МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»

Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
_____ Г.И. Радченко
«___»_____ 2020 г.

Веб-приложение для организации мероприятий по проведению настольных игр

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

Руководитель работы,
к.т.н., доцент каф. ЭВМ
_____Е.С. Ярош
«___»_____2020 г.

Автор работы,
студент группы КЭ-405
_____А.И. Гоглачев
«___»_____2020 г.

Нормоконтролёр,
ст. преп. каф. ЭВМ
_____ С.В. Сяськов
«___»_____2020 г.

Челябинск-2020

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____ Г.И. Радченко
«___»_____2020 г.

ЗАДАНИЕ

на выпускную квалификационную работу бакалавра

студенту группы КЭ-405
Гоглачеву Андрею Игоревичу
обучающемуся по направлению
09.03.01 «Информатика и вычислительная техника»

1. **Тема работы: «**Веб-приложение для организации мероприятий по проведению настольных игр» утверждена приказом №627 по университету от 24.04.2020

2. **Срок сдачи студентом законченной работы:** 1 июня 2020 г.

3. **Исходные данные к работе:**

Опции, подлежащие реализации:

– создание мероприятий по настольным играм;
– поиск мероприятий по фильтру и картам;
– поиск людей по интересам;
– обсуждение предстоящего мероприятия в чате;
– возможность связаться с пользователями с помощью личных сообщений;

– система администрирования приложения;
– интеграция с социальными сетями.

4. **Перечень подлежащих разработке вопросов:**

– анализ современного подхода к организации мероприятий по настольным играм;
– обзор аналогов;
– выбор технологий для реализации веб-приложения;
– проектирование базы данных;
– разработка приложения, выполняющего поставленную задачу;
– тестирование разработанного приложения.

5. **Дата выдачи задания:** 1 декабря 2019 г.

Руководитель работы _____/*Е.С. Ярош*/

Студент _____/*А.И. Гоглачев*/

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Введение и обзор литературы	01.03.2020	
Разработка модели, проектирование	01.04.2020	
Реализация системы	01.05.2020	
Тестирование, отладка, эксперименты	15.05.2020	
Компоновка текста работы и сдача на нормоконтроль	24.05.2020	
Подготовка презентации и доклада	30.05.2020	

Руководитель работы _____/*Е.С. Ярош* /

Студент _____/*А.И. Гоглачев* /

Аннотация

А.И. Гоглачев. Разработка веб-приложения для организации мероприятий по проведению настольных игр. – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШЭКН; 2020, 125 с., 40 ил., библиогр. список – 15 наим.

В рамках выпускной квалификационной работы было разработано веб-приложение для организации мероприятий по настольным играм. Были проанализированы существующие системы и выявлены их недостатки. Это обусловило необходимость данной разработки. Рассмотрены основные технологии, применяющиеся в разработке веб-приложений, и выбраны наиболее подходящие для данного проекта.

Были проведены проектирование, разработка и тестирование программного комплекса.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В настоящее время настольные игры в России набирают всё большую популярность. Большинство таких игр требует участия более чем двух игроков, и в большинстве случаев люди используют интернет в качестве средства коммуникации для организации встреч, поэтому существует необходимость в разработке веб-приложения для организации мероприятий по настольным играм.

Актуальность темы обусловлена тем, что основным методом коммуникации игроков являются социальные сети, которые обладают рядом недостатков, поэтому данный сервис будет нацелен на их устранение. Целью данной выпускной квалификационной работы является разработка приложения для упрощения коммуникации между людьми в вопросах о сборах с целью сыграть в настольные игры.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Рассмотреть существующие на данный момент системы, связанные с поставленной задачей.
2. Выполнить анализ преимуществ и недостатков найденных систем, применить полученные результаты в процессе проектирования и разработки.
3. Составить требования к основному функционалу системы.
4. Выбрать методы и средства реализации проекта.
5. Разработать схему базы данных.
6. Выполнить программную реализацию приложения.
7. Выполнить тестирование готового продукта.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. ОБЗОР АНАЛОГОВ

В настоящее время существуют веб-приложения для поиска мероприятий, связанных с настольными играми. Однако все они имеют некоторые недостатки, чаще всего, связанные с отсутствием возможности показа мест проведения, что затрудняет поиск подходящих мероприятий, а также может привести к неточностям. При этом большинство этих приложений имеют только англоязычную версию.

В соответствии с заданием, требуется реализация следующего функционала:

1. Создание мероприятий по настольным играм.
2. Поиск людей по интересам.
3. Возможность связаться с пользователями с помощью личных сообщений.
4. Поиск мероприятий по фильтру и картам.
5. Обсуждение предстоящего мероприятия в чате.

Рассмотрим наиболее популярные решения в данной предметной области, проведём анализ их преимуществ и недостатков.

1. Roll for Group [1]

Сервис предоставляет функционал по организации и поиску мероприятий с настольными играми. Также он позволяет находить других игроков по играм, в которые они хотят сыграть, и связываться с ними при помощи личных сообщений. Сообщество приложения небольшое, и в нём нет русскоязычных пользователей, что, скорее всего, связанно с отсутствием русской локализации. Также в процессе рассмотрения было замечено несколько некритических ошибок в работе приложения.

Преимущества:

– простой и удобный интерфейс для создания, поиска мероприятий;
– функция поиска отдельных людей;
– наличие системы уведомлений.

Недостатки:

– отсутствие поиска мероприятий на карте;
– наличие небольших ошибок в процессе работы.

2. GameTree [2]

Данное приложение позволяет пользователю искать людей для совместной игры и создавать игровые сессии. Несмотря на то, что предпочитаемыми играми можно указать настольные, функционал приложения направлен на компьютерные игры: отсутствует возможность выбора места проведения, его можно только указать в описании. Данное веб-приложение плохо подходит для решения поставленной задачи.

Преимущества:

– наличие русской локализации;
– наличие мобильных версий приложения.

Недостатки:

– отсутствие поиска мероприятий на карте;
– недостаточный функционал для работы с настольными играми.

3. GameFor [3]

Сервис позволяет создавать и искать мероприятия по настольным играм. Также пользователь имеет возможность искать других игроков. На сервисе мало параметров для фильтрации и поиска мероприятий и отсутствует система уведомлений.

Преимущества:

– возможность поиска местных магазинов настольных игр;
– наличие мобильных версий приложения.

Недостатки:

– отсутствие поиска мероприятий на карте;
– отсутствие фильтрации мероприятий по играм (только по видам игр);
– отсутствие возможности писать личные сообщения пользователям;
– отсутствие поиска игроков.

4. Meetup [4]

Сервис для поиска групп людей по интересам. Позволяет создавать группы по различным тема, в том и чсле и настольным играм, в каждой группе имеется возможность организовывать мероприятия.

Преимущества:

– возможность отображения мероприятий в календаре;
– наличие мобильных версий приложения.

Недостатки:

– отсутствие поиска мероприятий на карте;
– отсутствие фильтрации мероприятий.

1.2. АНАЛИЗ ОСНОВНЫХ ТЕХНОЛОГИЧЕСКИХ РЕШЕНИЙ

1.2.1. База данных

Чтобы хранить необходимую для работы приложения информацию требуется база данных.

База данных – это упорядоченный набор информации или данных, которые обычно хранятся в электронном виде в компьютерной системе. База данных обычно управляется системой управления базами данных (СУБД). Данные в наиболее распространенных типах современных баз данных обычно формируются в виде строк и столбцов в ряде таблиц, чтобы обеспечить эффективность обработки и запросов данных. Затем можно легко получать доступ к данным, управлять ими, изменять, обновлять, контролировать и упорядочивать. В большинстве баз данных для записи и запросов данных используется язык структурированных запросов (SQL).

СУБД служит интерфейсом между базой данных и пользователями или программами, предоставляя пользователям возможность получать и обновлять информацию, а также управлять ее упорядочением и оптимизацией. СУБД также упрощает контроль и управление базами данных, позволяя выполнять различные административные операции, такие как мониторинг производительности, настройка, а также резервное копирование и восстановление. [5]

Рассмотрим два наиболее распространенных вида баз данных: реляционные и нереляционные.

Реляционная база данных – это тип базы данных, которая хранит и предоставляют доступ к связанным друг с другом данным. Она основана на реляционной модели – интуитивно понятном и простом способе представления данных в таблицах. В такой базе данных каждая строка в таблице представляет

собой запись с уникальным идентификатором, называемым ключом. Столбцы таблицы содержат атрибуты данных, и каждая запись обычно содержит значение для каждого атрибута, упрощая установку отношений между данными[6].

Достоинства:

- наиболее простой и интуитивно понятный для пользователя способ представления данных;
- строгие правила проектирования базируются на математической модели;
- соответствие требованиям ACID (атомарность, согласованность, изолированность, надежность).

Недостатки:

- не дает преимуществ при работе со слабо связанными данными;
- более низкая по сравнению с нереляционными базами данных скорость доступа;
- трудоемкая разработка базы данных.

Нереляционные базы данных (NoSQL) не используют табличную схему данных. В этих базах данных применяется модель хранения, оптимизированная под конкретные требования типа хранимых данных. Для доступа к данным и управления ими применяются различные модели данных, в том числе документная, графовая, поисковая с использованием пар «ключ-значение» и хранением данных в памяти. Базы данных таких типов оптимизированы для приложений, которые работают с большим объемом данных, нуждаются в низкой задержке и гибких моделях данных. Все это достигается путем смягчения жестких требований к непротиворечивости данных, характерных для других типов БД [7].

Достоинства:

- хранение больших объёмов неструктурированной информации, NoSQL не накладывает ограничений на типы хранимых данных, при необходимости в процессе работы можно добавлять новые типы данных;
- NoSQL-базы данных лучше масштабируются за счет того, что их можно расширять горизонтально, т.е. подключая новые сервера к СУБД;
- более быстрая разработка базы данных.

Недостатки:

- все NoSQL системы имеют собственный язык запросов, что привязывает разработанный проект к одной СУБД;
- NoSQL не может обеспечить полную поддержку требований ACID.

Вывод: для реализации функционала данного проекта требуется связь между данными (например, для удаления пользователя и всей информации, связанной с ним), также разработанный проект не будет требовать большой масштабируемости. В итоге целесообразно выбрать реляционную базу данных.

Так как разрабатываемое приложение имеет социальную направленность, будут рассмотрены свободно распространяемые СУБД. Формат веб-приложения требует решения проблемы хостинга, причем с минимальными затратами. Это также сужает круг рассматриваемых СУБД.

SQLite – это автономная файловая открытая СУБД. Её главными особенностями являются мобильность и высокая производительность даже в средах с нехваткой памяти. Её транзакции совместимы с ACID. SQLite описывается как «безсерверная» база данных. Большинство механизмов реляционных баз данных реализованы в виде серверного процесса, с которым другие программы взаимодействуют посредством запросов. Но в SQLite любой процесс, который обращается к базе данных, читает и записывает данные

непосредственно в файл базы данных. Это упрощает процесс установки SQLite, так как устраняется необходимость в настройке сервера. SQLite является бесплатным программным обеспечением с открытым исходным кодом, и для его использования не требуется специальной лицензии [8].

Достоинства:

- библиотека SQLite является легковесной и полностью автономной, для нее не требуется никаких внешних зависимостей;
- для работы с базой данных не требуется настраивать и запускать серверный процесс, что упрощает интеграцию с приложением;
- вся база данных хранится в одном файле, что делает ее легко переносимой.

Недостатки:

- только один процесс может вносить изменения в базу данных в любой момент времени, что ограничивает параллелизм;
- в SQLite отсутствует управление пользователями, что делает его неподходящим для приложений, где требуется несколько пользователей с разными правами доступа;
- ядро базы данных, использующей сервер, обеспечивает более надежную защиту от ошибок, чем SQLite.

MySQL обеспечивает очень быстрый, многопоточный, многопользовательский и надежный сервер баз данных SQL (Structured Query Language). Программное обеспечение MySQL имеет двойную лицензию: пользователи могут выбрать использование программного обеспечения MySQL в качестве продукта с открытым исходным кодом на условиях стандартной общественной лицензии GNU или могут приобрести стандартную

коммерческую лицензию у Oracle. MySQL имеет широкое распространение и поддерживает веб-сайты и приложения [9].

Достоинства:

– популярность MySQL означает, что имеется множество информации об установке и управлению базой данных, также существует множество инструментов, упрощающих работу с MySQL.

Недостатки:

– MySQL был разработан для скорости и простоты использования, поэтому не соответствует полностью стандарту SQL, он имеет определенные функциональные ограничения;
– бесплатной версией сообщества могут пользоваться только проекты с открытым исходным кодом, а некоторые функции и плагины доступны только для проприетарных версий.

PostgreSQL – свободно распространяемая объектно-реляционная СУБД. Это означает, что она также включает в себя такие функции, как наследование таблиц и перегрузка функций. PostgreSQL поддерживает обширный список типов данных, кроме стандартных типов данных в нем также присутствуют: uuid, денежный, перечисляемый, геометрический, бинарный тип данных, сетевые адреса, битовые строки, xml, json, массивы.

Достоинства:

– PostgreSQL стремится к более строгому соблюдению стандартов SQL, чем другие СУБД. PostgreSQL поддерживает 160 из 179 функций, необходимых для полного соответствия требованиям SQL;
– PostgreSQL свободно распространяется и его использование не накладывает никаких ограничений на проект;

– поддержка пользовательских типов данных и их поведения делает данную СУБД более гибкой.

Недостатки:

– PostgresSQL используется не так широко, как MySQL, это означает, что для данной СУБД существует меньше сторонних инструментов, помогающих в управлении базой данных;
– для каждого нового клиентского соединения PostgreSQL разветвляется новый процесс, требующий примерно 10 МБ памяти, что довольно значительно для базы данных с большим количеством подключений.

Вывод: в ходе выбора можно сразу отклонить вариант SQLite, данную СУБД можно использовать в процессе разработки проекта из-за ее простоты, но для дальнейшей эксплуатации приложения нужна более производительная база данных.

PosgreSQL является хорошим выбором, но так как ее функционал, связанный с объектными базами данных, не является необходимом в данном проекте, остановим выбор на MySQL за счет ее производительности, разнообразия сторонних инструментов для управления базой данных и большого количества руководств по работе с этой СУБД.

1.2.2. Серверная часть приложения

1.2.2.1. Выбор языка программирования

Рассмотрим наиболее популярные языки программирования для серверной части веб-приложения. Проанализируем их преимущества и недостатки для выбора оптимального варианта.

PHP – язык программирования, разработанный специально для написания серверной части веб-приложения, с синтаксисом во многом похожим на C и Java. Основная задача языка – поддержка динамических HTML-страниц, для чего разработчики на PHP пишут скрипты. В результате работы такого скрипта пользователю будет отправлен HTML с необходимыми данными. На текущий момент PHP является одним из наиболее распространенных языков веб-программирования, по данным W3Techs [10] на 2020 год PHP используется в 78.8% всех известных сайтов.

Преимущества:

- простота освоения;
- развитая поддержка баз данных;
- богатая история языка обеспечивает его высокую стабильность и отсутствие критических ошибок.

Недостатки:

- открытость исходного кода интерпретатора языка упрощает поиск уязвимостей злоумышленниками;
- большинство PHP фреймворков предназначены для разработки традиционных веб-сайтов и плохо подходят для одностраничных приложений;
- возможность смешивания HTML и PHP понижает читабельность кодовой базы.

Python – универсальный язык программирования, используемый также и в веб-разработке. Python поддерживает большинство актуальных платформ. Язык был разработан с целью быть максимально понятным при чтении и быть простым в изучении. Самые популярные фреймворки для веб-разработки на Python позволяют разработать веб-приложения full-stack, без использования дополнительных библиотек, другие же обладают минимально необходимыми

функциями, предоставляя разработчику полную свободу в выборе остальных средств разработки.

Преимущества:

– простой синтаксис делает Python одним из самых простых языков в изучении;
– поддержка асинхронного кода, что важно для веб-приложений.

Недостатки:

– низкая производительность из-за того, что Python являет интерпретируемым, а не компилируемым языком;
– недостаток средств доступа к базам данных.

JavaScript – изначально язык был создан с целью разработки скриптов для HTML-страниц. Сейчас область применения JavaScript вышла за пределы браузеров, с применением «движка» JavaScript может быть запущен сервере или на любом другом устройстве. Для серверов для запуска JavaScript используется среда выполнения Node.js, основанная на движке Google Chrome – V8, который компилирует JavaScript код в машинный код. Node.js использует событийно-ориентированную модель и неблокирующую ввод/вывод архитектуру, что делает его легковесным и эффективным [11].

В качестве языка для разработки серверной части приложения выберем JavaScript, так как это упрощает переход между разработкой клиентской и серверной частей. Также богатая «экосистема» языка предоставляет большое количество руководств по разработке и библиотек.

1.2.2.2. Выбор фреймворка

Рассмотрим наиболее популярные фреймворки для Node.js: Express, Koa и Sails [12].

1. Express

   Достоинства:

– низкий порог вхождения;
– возможность быстрой разработки приложений;
– высокая степень настраиваемости за счет поддержки большого числа библиотек;
– широкое сообщество и высокая популярность означает наличие большого числа информации про работу с фреймворком.

   Недостатки:

– необходимость вручную настраивать безопасность приложения;
– читабельность кода может снизится при наличии большого количества callback-функций.

2. Koa

   Достоинства:

– упрощенная обработка ошибок;
– встроенная поддержка ключевых слов async/await;

   Недостатки:

– отсутствие совместимости с другими middleware-функциями;
– по умолчанию отсутствуют методы маршрутизации;
– небольшое сообщество разработчиков.

3. Sails

   Преимущества:

– встроенная поддержка большинства популярных баз данных;
– хорошая поддержка Socket.io.

   Недостатки:

– низкая скорость разработки приложений;
– плохая производительность.

В качестве фреймворка для серверной части приложения выберем Express за счет его простоты и гибкости в разработке, а также наличию большого количества материалов про разработку на этом фреймворке.

1.2.3. Выбор фреймворка для клиентской части приложения

Рассмотрим несколько фреймворков для frontend.

1. Angular – основанный на TypeScript фреймворк, поддерживаемый Google. Он предназначен для отделения логики приложения от манипуляций с DOM (Document Object Model или объектная модель документа) и нацелен на динамическое обновление страниц. В нём была представлена концепция привязки данных, которая подразумевала автоматическое обновление представления при изменении модели (данных) и наоборот. Кроме того, была представлена идея директив, которая позволила реализовывать собственные HTML-теги.

   Преимущества:

– полная поддержка TypeScript;
– архитектура, специально созданная для больших приложений;
– взаимозависимость функций, в виду их связанности с компонентами и модулями.

   Недостатки:

– разнообразие различных структур (Injectables, Components, Pipes, Modules и т.д.) усложняет изучение по сравнению с React и Vue.js, которые имеют только «Component»;
– относительно низкая производительность.

2. ReactJS – это библиотека JavaScript, используемая для создания пользовательских интерфейсов. React был создан компанией Facebook.

React представляет собой хороший инструмент для создания масштабируемых веб-приложений, особенно при разработке одностраничного приложения (SPA).

Преимущества:

– использование компонентной архитектуры;
– множество документации и онлайн-ресурсов.

Недостатки:

– React использует JSX, который сложен в изучении;
– необходимость в средствах сборки для совместимости с другими библиотеками.

3. Vue.js – JavaScript фреймворк с открытым исходным кодом, предназначенный для создания пользовательских интерфейсов. На данный момент он поддерживается сообществом разработчиков, но не уступает Angrular и ReactJS. Vue является отличным выбор для разработки приложений с двухсторонней привязкой данных.

Преимущества:

– Vue намного легче изучить и использовать по сравнению с другими фреймворками;
– использование виртуального DOM, что повышает производительность;
– возможность создания однофайловых компонентов, что упрощает организацию проекта и улучшает читабельность кода.

Недостатки:

– по сравнению с остальными популярными фреймворками, Vue обладает гораздо меньшим количеством ресурсов для разработки и обучения;

Для реализации проекта используем фреймворк Vue.js из-за его простоты его изучения.

1.3. Выбор картографического сервиса

Картографический сервис – это система, предоставляющая пространственные данные в виде интерактивной карты. Такая система обеспечивает веб-доступ к информации при помощи интерфейсов прикладного программирования (API). В настоящее время на российском рынке наиболее известны и распространены следующие картографические и справочные сервисы:

– Яндекс.Карты;
– Google Maps;
– 2ГИС.

Сравнение рассматриваемых сервисов приведено в таблице 1.1.

Таблица 1.1 – Сравнение картографических сервисов [13]

Критерий	**Яндекс.Карты**	**Google Maps**	**2ГИС**
Покрытие	Карта всего мира (но наиболее проработаны карты России, Украины, Белоруссии и Казахстана, а также Европы и Северной Америки)	Карта всего мира (но хорошо прорисованы только наиболее крупные города Северной Америки, Европы, России и др.)	Россия и несколько городов в 9 странах (всего около 398 городов)
Условия использования API	Бесплатно для использования в открытых некоммерческих неигровых проектах, не предназначенных для мониторинга и диспетчеризации. Обязательна регистрация и получение ключа.	Полностью бесплатное использование отсутствует, можно использовать бесплатно ежемесячно на сумму 200$	Бесплатно для использования в открытых некоммерческих проектах, не направленных на построение маршрутов. Обязательна регистрация и получение ключа.

Продолжение таблицы 1.1

Критерий	Яндекс.Карты	Google Maps	2ГИС
Построение маршрутов	Построение нескольких вариантов маршрута на автомобиле (с учетом пробок), общественным транспортом, пешком. Расчёт предположительного времени в пути. Проигрывает Google.Maps в качестве построения маршрута.	Построение нескольких вариантов маршрута на автомобиле (с учетом пробок), общественным транспортом, пешком, на велосипеде и даже самолетом. Расчёт предположительного времени в пути.	Построение нескольких маршрутов на автомобиле, общественном транспорте, пешком с расчётом времени на путь.
Режимы отображения карты	Режимы «Схема», «Спутник», «Гибрид», панорамы некоторых городов	Режимы «Схема» и «Спутник», панорамы отдельных городов	Режим «Схема»
Ограничения количества запросов при бесплатном использовании API	Число запросов к сервисам геокодирования, маршрутизации и панорам Яндекса не должно превышать 25 000 в сутки.	Число загрузок карт не должно превышать превышает 25 000 в сутки.	Количество запросов к сервису ограничено предельной величиной 10 в секунду и (или) 10000 в месяц
Документация по использованию API	Документация очень подробная, с примерами использования большинства функций.	Документация достаточно подробная, но частично на английском языке.	Документация по использованию краткая

Продолжение таблицы 1.1

Критерий	Яндекс.Карты	Google Maps	2ГИС
Элементы управления	– Элементы для перетягивания карты, увеличения выделенной области, измерения расстояний. – Элемент изменения масштаба – Переключатель типа карты – Масштабная линейка – Обзорная карта – Поиск по карте – Пробки – Редактор маршрута – Пользовательские элементы управления	– Масштабирование карты – Выбор типа карты – Элемент управления Street View – Элемент управления Rotate для наклона и вращения – Элемент перехода в полноэкранный режим – Построение маршрутов – Пользовательские элементы управления	– Управление – Масштаб – Линейка – Отображение слоя пробок – Кнопка полноэкранного отображения карты – Определение месторасположения пользователя
Средства для вывода большого количества данных	Кластеризация; Технология активных областей; Технологии ObjectManager, LoadingObjectManager, RemoteObjectManager	Кластеризация маркеров; Технология setTimeout для последовательного вывода маркеров на карту.	Кластеризация объектов

В качестве картографического сервиса будут использованы Яндекс.Карты, так как 2ГИС бесплатно не предоставляет достаточный функционал для работы приложения, а у Google Maps отсутствует возможность полностью бесплатного использования.

1.4. ВЫВОД

Из обзора аналогов можно сделать вывод, что ни один из них не поддерживает весь набор требований задания, поэтому данная разработка является актуальной.

Основные технические решения:

Разработка базы данных будет осуществляться для реляционной СУБД MySQL при помощи инструмента визуального проектирования баз данных MySQL Workbench.

Серверная часть приложения будет разработана на языке JavaScript для среды выполнения Node.js с использованием фреймворка Express.

Для разработки клиентской части приложения будет применен JavaScript фреймворк Vue.js, также будут использоваться язык разметки HTML и язык описания внешнего вида документа CSS.

2. ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ

2.1. ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

2.1.1. Основные требования к функциональности системы

1. Регистрация и авторизация пользователей.
2. Авторизация через социальную сеть Вконтакте.
3. Редактирование профиля пользователя.
4. Хранение данных пользователей в базе данных.
5. Создание мероприятий с возможностью выбора даты, времени, места проведения (ввод адреса в поле или выбор на карте) и настольных игр (поле с автозавершением по данным с сайта tesera.ru). Создавать мероприятия может любой зарегистрированный пользователь.
6. Запись на доступные мероприятия.
7. Организация чатов в реальном времени для каждого мероприятия.
8. Поиск мероприятий по фильтрам и на карте.
9. Поиск других пользователей по фильтрам.
10. Организация системы администрирования сервиса.
11. Редактирование мероприятий.
12. Организация системы личных сообщений.
13. Настройка приватности мероприятий (публичное, по заявкам).
14. Организация системы связи пользователей с администраторами.
15. В системе должны присутствовать следующие типы пользователей:
    1. Гость
    2. Зарегистрированный пользователь, который подразделяется на:
        1) участника мероприятия
        2) организатора мероприятия

3. Администратор

2.1.2. Требования к функционалу системы администрирования

1. Администрирование пользователей:

    а) блокирование пользователя с возможностью ручного или автоматического снятия блокировки.

    б) удаление пользователя со всеми данными, связанными с ним.
2. Администрирование мероприятий:

    а) блокирование мероприятия.

    б) удаление мероприятия.
3. Администрирование сообщений:

    а) удаление сообщения.

2.1.3. Требования к системе уведомлений

Приложение должно обладать следующим минимальным набором уведомлений:

1. Получение личного сообщения.
2. Получения заявки на участие в мероприятии.
3. Предупреждение об отмене мероприятия.
4. Уведомление о блокировке профиля с причиной.
5. Уведомление о блокировке мероприятия для организатора (с указанием причины) и участников.

2.2. НЕФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

2.2.1. Требования к пользователю

1. Пользователь должен указать отображаемое имя, адрес электронной почты и пароль при регистрации.
2. После регистрации пользователь обязан подтвердить свой адрес электронной почты. При отсутствии подтверждения профиль будет заблокирован и в дальнейшем удалён.

2.2.2. Требования к системе безопасности

Система должна обладать следующими средствами обеспечения безопасности:

1. Пользователь при входе в систему обязан указывать логин и пароль.
2. Пользователь при регистрации в системе обязать указывать пароль два раза (для подтверждения).
3. Пароль должен храниться в зашифрованном виде.

3. ПРОЕКТИРОВАНИЕ

3.1. АРХИТЕКТУРА ПРЕДЛАГАЕМОГО РЕШЕНИЯ

Приложение будет построено по следующей архитектуре: на клиентской стороне пользователь будет взаимодействовать с SPA (Single Page Application или одностраничное приложение), данные для которого будут получены при помощи REST API на сервере.

REST API будет обеспечивать взаимодействие приложения и сервера при помощи протокола HTTP. При такой архитектуре взаимодействие с сервером осуществляется четырьмя операциям:

- получение данных с сервера;
- добавление новых данных на сервер;
- изменение данных;
- удаление данных.

Каждой из этих операций соответствует свой метод HTTP-запроса: GET, POST, PUT и DELETE соответственно.

SPA – веб-приложение, которое работает на одной HTML-странице. Все необходимые шаблоны страниц, JavaScript и CSS файлы подгружаются при первой загрузке, в дальнейшем взаимодействие приложения и сервера сводится к минимуму. Большая часть работы приложения производится на устройстве пользователя, а сервер только отправляет необходимые данные, обычно в формате JSON.

На рисунке 3.1 представлена схема взаимодействия REST API с SPA.

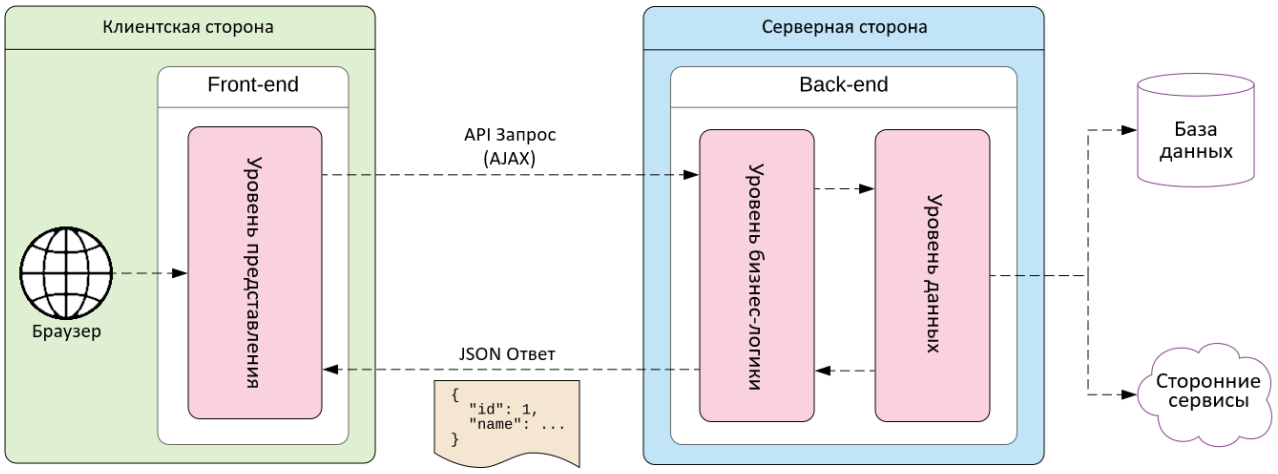

Рисунок 3.1 – взаимодействие REST API и SPA

В одностраничных приложениях отображение полученных пользователем данных осуществляется динамически при помощи AJAX (Asynchronous JavaScript And XML), который представляет собой набор методов разработки и технологий и позволяет веб-приложению совершать свою работу асинхронно, то есть при обновлении конкретной части страницы не будет необходимости перезагружать её всю.

Основным преимуществом такой архитектуры является то, что одностраничные приложения создают у пользователя впечатление работы с десктоп приложением из-за практически моментальной реакции на действия, чего нельзя добиться обычным подходом. SPA обеспечивает максимальную интерактивность с пользователем.

Другим преимуществом архитектуры можно назвать понижение нагрузки на сервер, так как нет необходимости каждый раз отправлять пользователю, созданную по шаблону, новую разметку страницы, отправляются только самые необходимые данные. Также REST API не привязывается к одному приложению, данные с него могут использоваться, как в веб, так в нативных версиях.

3.2. ДИАГРАММА ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ

Можно выделить следующие типы пользователей приложения:

– неавторизованный пользователь;

– авторизованный пользователь;

– администратор.

UML диаграмма вариантов использования разрабатываемого проекта представлена на рисунке 3.2.

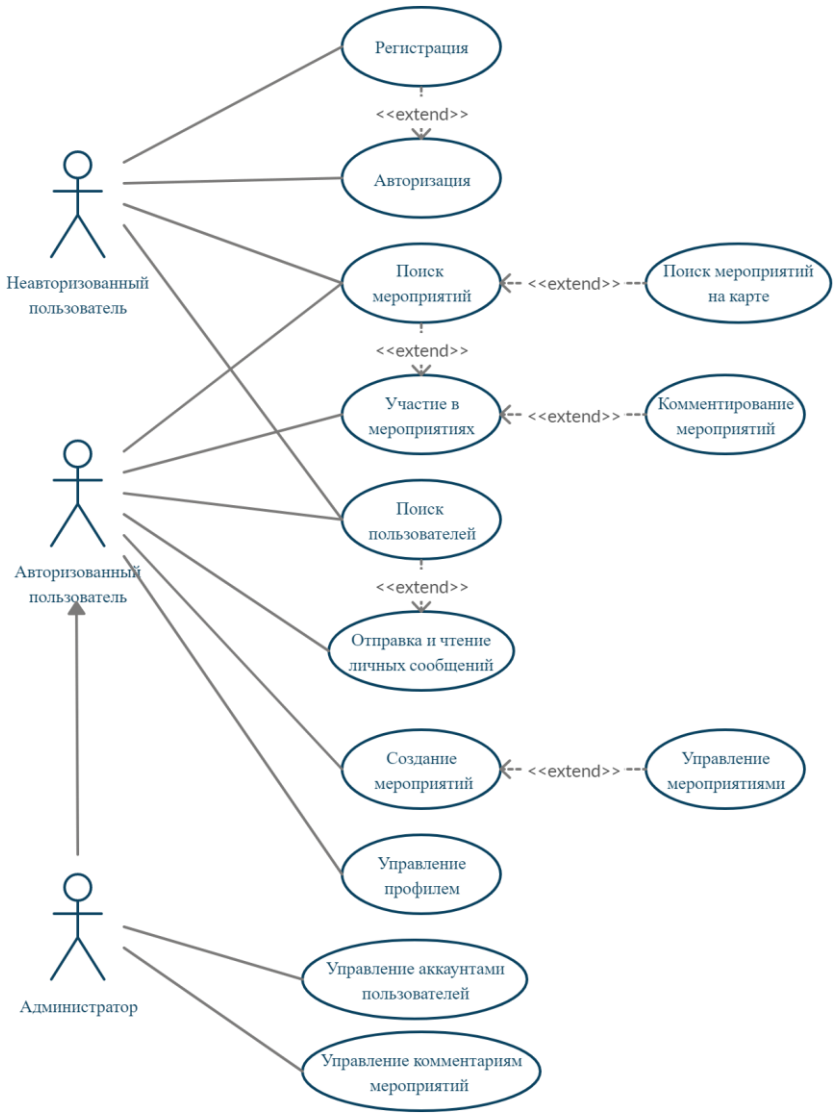

Рисунок 3.2 – Диаграмма вариантов использования

3.3. ОПИСАНИЕ ДАННЫХ

Схема базы данных, разработанная в MySQL Workbench, представлена на рисунке 3.3.

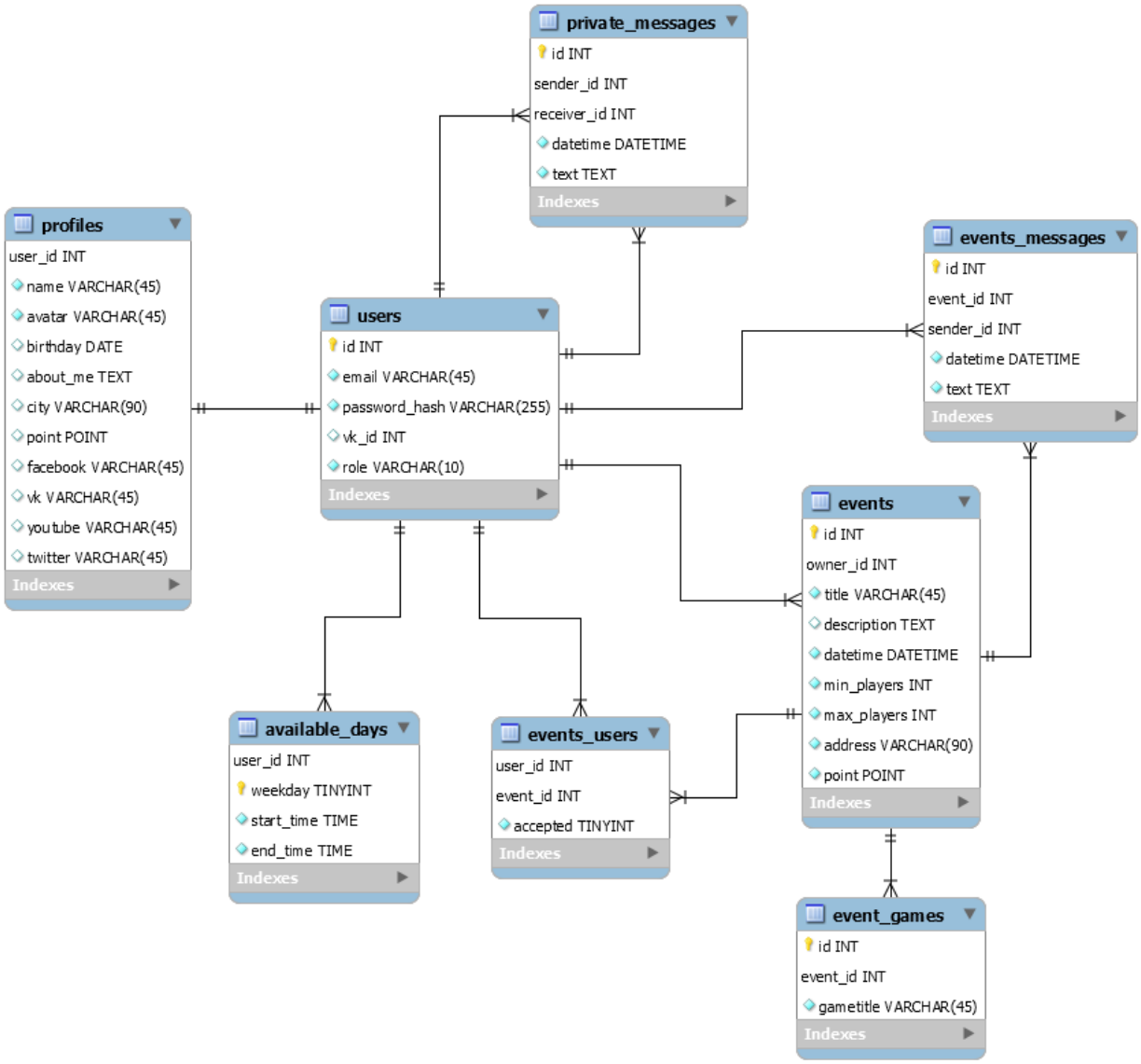

Рисунок 3.3 – Схема базы данных

Таблицы:

users – содержит информацию для аутентификации пользователей, столбцы приведены в таблице 3.1.

Таблица 3.1 – Таблица “users”

Название столбца	**Тип данных**	**Описание**
id	INT	Идентификатор пользователя, первичный ключ
email	VARCHAR	Почта пользователя
password_hash	VARCHAR	Хэш пароля пользователя
vk_id	INT	Идентификатор пользователя в социальной сети Вконтакте
role	VARCHAR	Роль пользователя

profiles – содержит информацию профиля пользователя, столбцы приведены в таблице 3.2.

Таблица 3.2 – Таблица “profiles”

Название столбца	**Тип данных**	**Описание**
user_id	INT	Идентификатор пользователя, внешний ключ из таблицы “users“
name	VARCHAR	Имя пользователя
avatar	VARCHAR	Имя файла изображения профиля пользователя
birthday	DATE	Дата дня рождения пользователя
about_me	TEXT	Информация пользователя о себе
city	VARCHAR	Название города, в котором проживает пользователь

Продолжение таблицы 3.2

point	POINT	Координаты города, в котором проживает пользователь
facebook	VARCHAR	Ссылка на профиль пользователя в социальной сети Facebook
vk	VARCHAR	Ссылка на профиль пользователя в социальной сети Вконтакте
youtube	VARCHAR	Ссылка на профиль пользователя на видеохостинге YouTube
twitter	VARCHAR	Ссылка на профиль пользователя в социальной сети Twitter

events – содержит информацию о мероприятиях, столбцы приведены в таблице 3.3.

Таблица 3.3 – Таблица “events”

Название столбца	**Тип данных**	**Описание**
id	INT	Идентификатор мероприятия, первичный ключ
owner_id	INT	Идентификатор организатора, внешний ключ из таблицы “users“
title	VARCHAR	Название мероприятия
description	TEXT	Описание мероприятия
datetime	DATETIME	Дата и время начала проведения
max_players	INT	Максимальное количество игроков
address	VARCHAR	Адрес проведения мероприятия
location	POINT	Координаты места проведения

events_users – содержит информацию о том, кто участвует в мероприятии, столбцы приведены в таблице 3.4.

Таблица 3.4 – Таблица “events_users”

Название столбца	**Тип данных**	**Описание**
user_id	INT	Идентификатор пользователя, внешний ключ из таблицы “users“
event_id	INT	Идентификатор мероприятия, внешний ключ из таблицы “events“
accepted	TINYINT	Статус пользователя: 0 – пользователь отправил заявку, 1 – пользователь принят на мероприятие

event_games – содержит информацию о настольных играх, выбранных для мероприятия, столбцы приведены в таблице 3.5.

Таблица 3.5 – Таблица “event_games”

Название столбца	**Тип данных**	**Описание**
event_id	INT	Идентификатор мероприятия, внешний ключ из таблицы “events“
gametitle	VARCHAR	Название настольной игры

private_messages – содержит личные сообщения пользователей, столбцы приведены в таблице 3.6.

Таблица 3.6 – Таблица “private_messages”

Название столбца	**Тип данных**	**Описание**
id	INT	Идентификатор сообщения, первичный ключ
sender_id	INT	Идентификатор пользователя отправителя, внешний ключ из таблицы “users“
receiver_id	INT	Идентификатор пользователя получателя, внешний ключ из таблицы “users“
datetime	DATETIME	Дата и время отправки
text	TEXT	Текст сообщения

events_messages – содержит комментарии пользователей к мероприятиям, столбцы приведены в таблице 3.7.

Таблица 3.7 – Таблица “events_messages”

Название столбца	**Тип данных**	**Описание**
id	INT	Первичный ключ
event_id	INT	Идентификатор мероприятия, внешний ключ из таблицы “event“
sender_id	INT	Идентификатор отправителя комментария, внешний ключ из таблицы “users“,
datetime	DATETIME	Дата и время отправки
text	TEXT	Текст комментария

available_days – содержит расписание пользователя, столбцы приведены в таблице 3.8.

Таблица 3.8 – Таблица “available_days”

Название столбца	Тип данных	Описание
user_id	INT	Идентификатор пользователя, внешний ключ из таблицы “users“
weekday	TINYINT	Номер для недели, первичный ключ
start_time	TIME	Начало промежутка свободного времени пользователя
end_time	TIME	Конец промежутка свободного времени пользователя

3.4. КОМПОНЕНТЫ SPA

Клиентским фреймворком, использующимся для реализации SPA, был выбран Vue.js. Для приложений, где весь фронтенд управляется JavaScript, для более эффективной разработки рекомендуется применять однофайловые компоненты. Это файлы с расширением .vue, содержащие HTML разметку компонента, его скрипты и стили. Их использование позволяет разбить всю структуру приложения на набор отдельных модулей, что в целом упрощает разработку.

Файлы можно разбить на страницы и компоненты: страницы используются для маршрутизации и определяют, что видит пользователь на каждом из маршрутов, компоненты – это составные части страницы, вынесенные в отдельные файлы, например, для повторного использования.

В таблице 3.9 содержатся файлы страниц, исходные коды файлов приведены в приложения Б, В, Г, Д, Е, Ж, К.

Таблица 3.9 – Файлы страниц

Название	**Назначение**
App.vue	Основная страницы приложения
EventsList.vue	Страница списка мероприятия
EventPage.vue	Страница мероприятия
Members.vue	Страница списка участников мероприятия
UsersList.vue	Страница списка пользователей
User.vue	Страница профиля пользователя
Settings.vue	Страница редактирования профиля пользователя
NewEventForm.vue	Страница создания нового мероприятия
ChatsList.vue	Страница списка чатов пользователя
Chat.vue	Страница чата

В таблице 3.10 содержатся файлы компонентов, исходные коды файлов приведены в приложения М, Н, П.

Таблица 3.10 – файлы компонентов

Название	**Назначение**
AppBar.vue	Компонент навигационной панели приложения
Avatar.vue	Компонент для смены изображения профиля пользователя
CitySearch.vue	Компонент текстового поля с автозаполнением для поиска городов

Продолжение таблицы 3.10

EventCard.vue	Компонент карточки мероприятия для отображения в списке
GameSearch.vue	Компонент текстового поля с автозаполнением для поиска игр
LocationPicker.vue	Компонент для выбора места проведения мероприятия на карте
LoginForm.vue	Компонент окна авторизации в приложении
RegisterForm.vue	Компонент окна регистрации в приложении
Snackbar.vue	Компонент всплывающих сообщений

4. РЕАЛИЗАЦИЯ

4.1. ФОРМА АВТОРИЗАЦИИ В ПРИЛОЖЕНИИ

При нажатии кнопки “войти” в навигационной панели или попытке пользователя получить доступ к недоступному для гостей маршруту откроется диалоговое окно авторизации, показанное на рисунке 4.1.

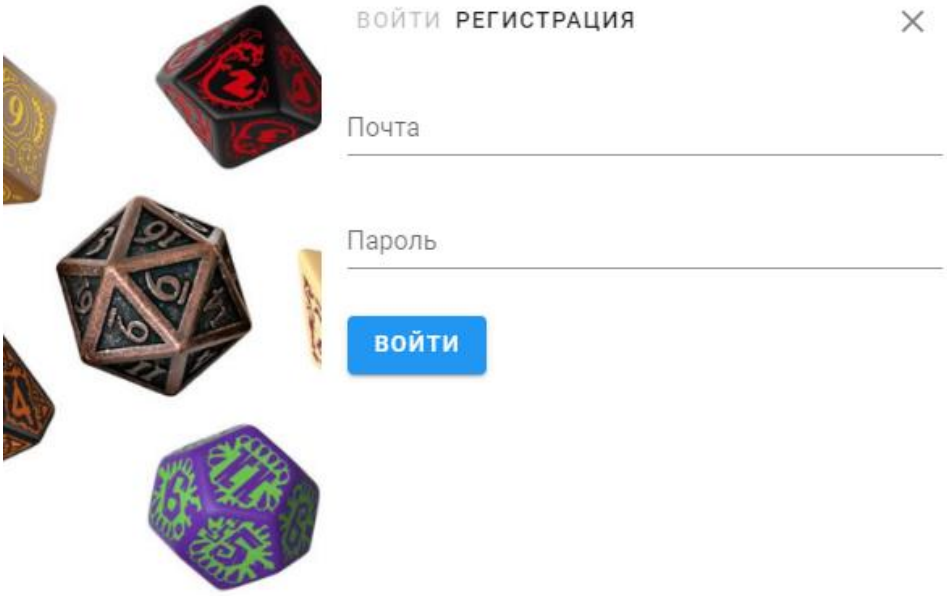

Рисунок 4.1 – Диалоговое окно авторизации

При нажатии кнопки “регистрация” окно переключится на форму регистрации, где необходимо обязательно заполнить поля имени, почты и пароля пользователя, пароль необходимо указать два раза. Электронная почта пользователя проверяется на корректность. Окно регистрации представлено на рисунке 4.2.

При регистрации пароль пользователя хэшируется при помощи хэш-функции bcrypt и сохраняется в базе данных поле “password_hash” таблицы “users”, сам пароль пользователя не хранится на сервере. Далее при попытке пользователя авторизоваться его пароль сверяется с хранимым хэшем. Данный

метод реализован при помощи библиотеки bcrypt.js и позволяет обезопасить пароли пользователей в случае получения доступа к базе данных злоумышленниками.

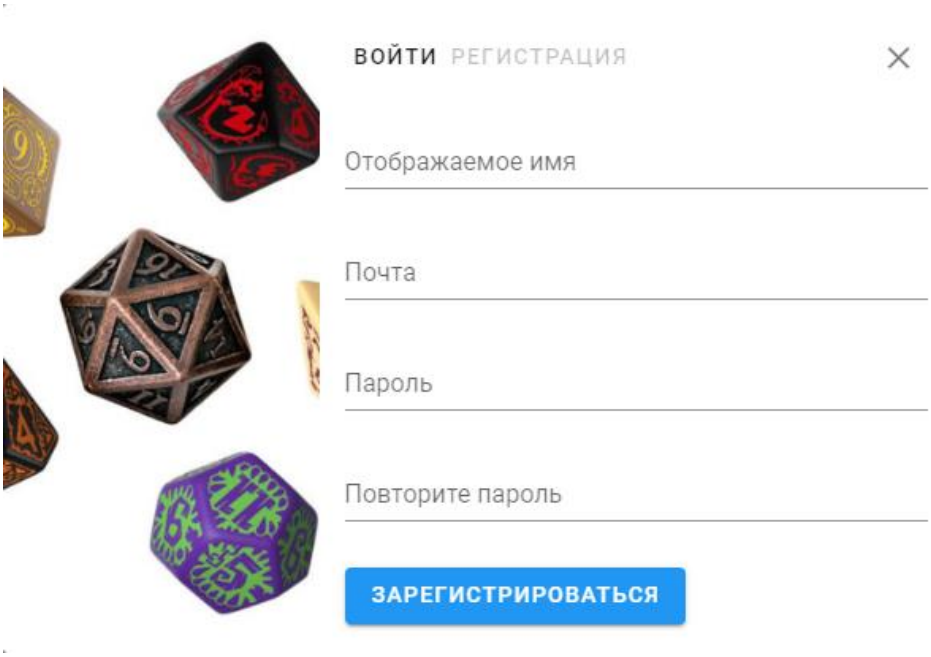

Рисунок 4.2 – Диалоговое окно регистрации

4.2. ОСНОВНОЕ ОКНО ПРИЛОЖЕНИЯ

Авторизированным пользователям доступны следующие маршруты приложения, ссылки на которые расположены в левой боковой панели:

- игровые сессии;
- создать сессию;
- поиск игроков;
- сообщения (видима только на устройствах с низкой шириной экрана).

В правой боковой панели, видимой только авторизированным пользователям, расположены все диалоги пользователя, нажатие на которые направит на соответствующие страницы с диалогами.

При попытке неавторизированного пользователя создать сессию, он будет его автоматически перенаправлен на окно авторизации. Проверка авторизации пользователя осуществляется просмотром наличия токена авторизации в хранилище состояний. Такое хранилище позволяет назначать данные, которые будут доступны всем компонентам приложения. Для фреймворка Vue.js глобальные состояния приложения реализуются при помощи библиотеки Vuex.

Несмотря на то, что пользователь может добавить токен самостоятельно, при попытке получить доступ к защищенному маршруту данные будут проверены на действительность. Основное окно приложение представлено на рисунке 4.3.

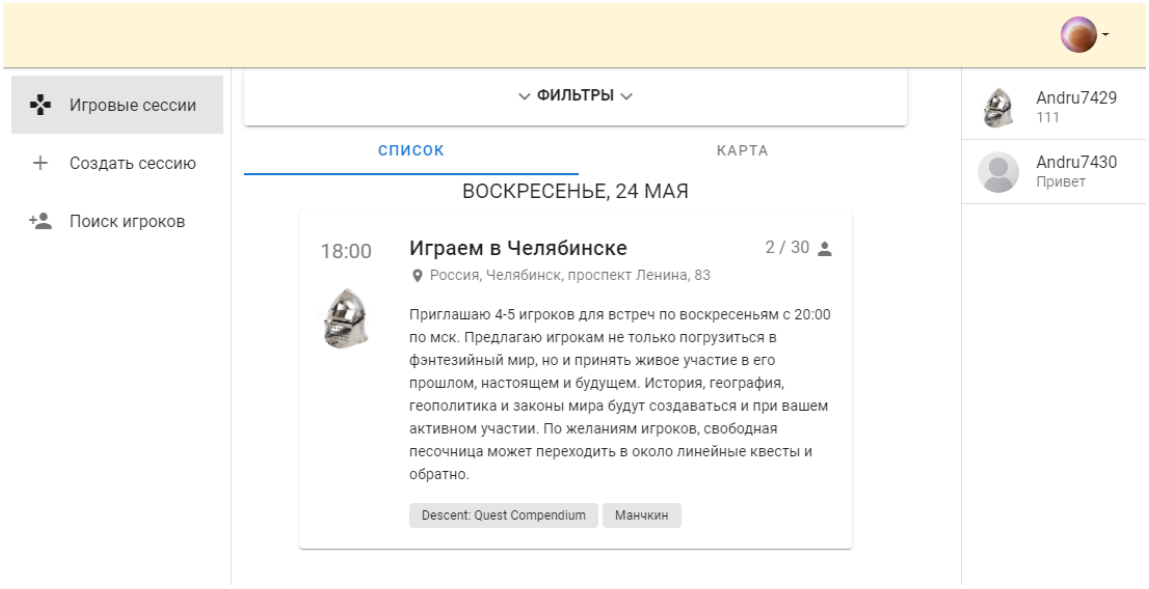

Рисунок 4.3 – Основное окно приложения

Проект реализован с адаптивным дизайном, что делает его применимым на устройствах с низким горизонтальным разрешением. Адаптивность реализована с помощью библиотеки Vuetify, которая предоставляет специальный компонент, позволяющий создавать разные разметки для устройств с различной шириной экрана. На рисунке 4.4 представлен вид приложения на мобильном устройстве.

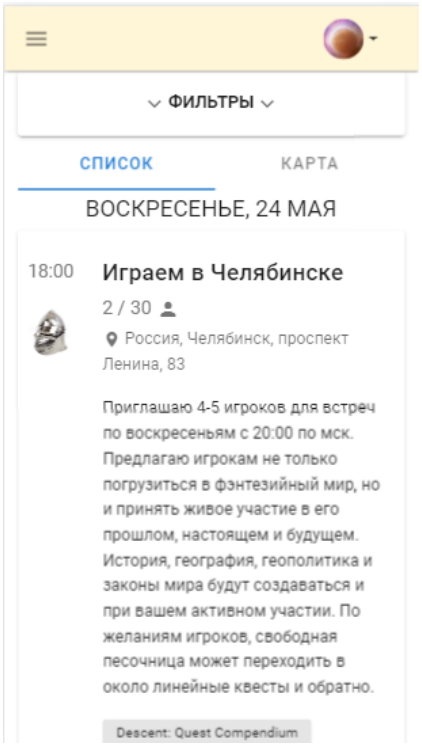

Рисунок 4.4 – Вид приложения на мобильном устройстве

По нажатию иконки с тремя горизонтальными линиями в правом верхнем углу можно открыть боковую панель навигации. Функционал правой боковой панели с диалогами дублируется отдельной страницей, доступной мобильными устройствам, Ссылка на эту страницу видима только на устройствах с низкой шириной экрана и содержит тот же компонент, что и в боковой панели (ChatsList.vue). Открытая боковая панель представлена на рисунке 4.5.

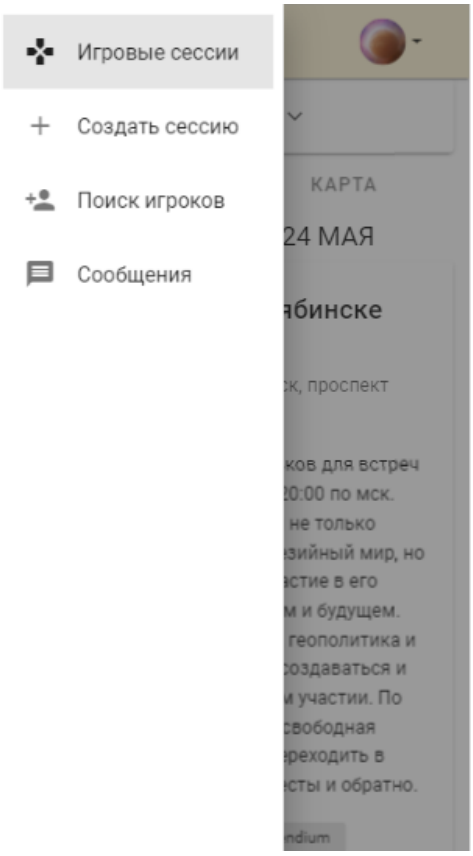

Рисунок 4.5 – Вид боковой панели на мобильном устройстве

4.3. СОЗДАНИЕ И ПОИСК МЕРОПРИЯТИЙ

Переход к форме создания мероприятия осуществляется нажатием кнопки “создать сессию” в боковой панели. В форме отслеживается заполнение обязательных полей, время и дата задаются через специальные меню. Форма представлена на рисунке 4.6. Исходный код страницы приведен в листинге Г.1 приложения Г.

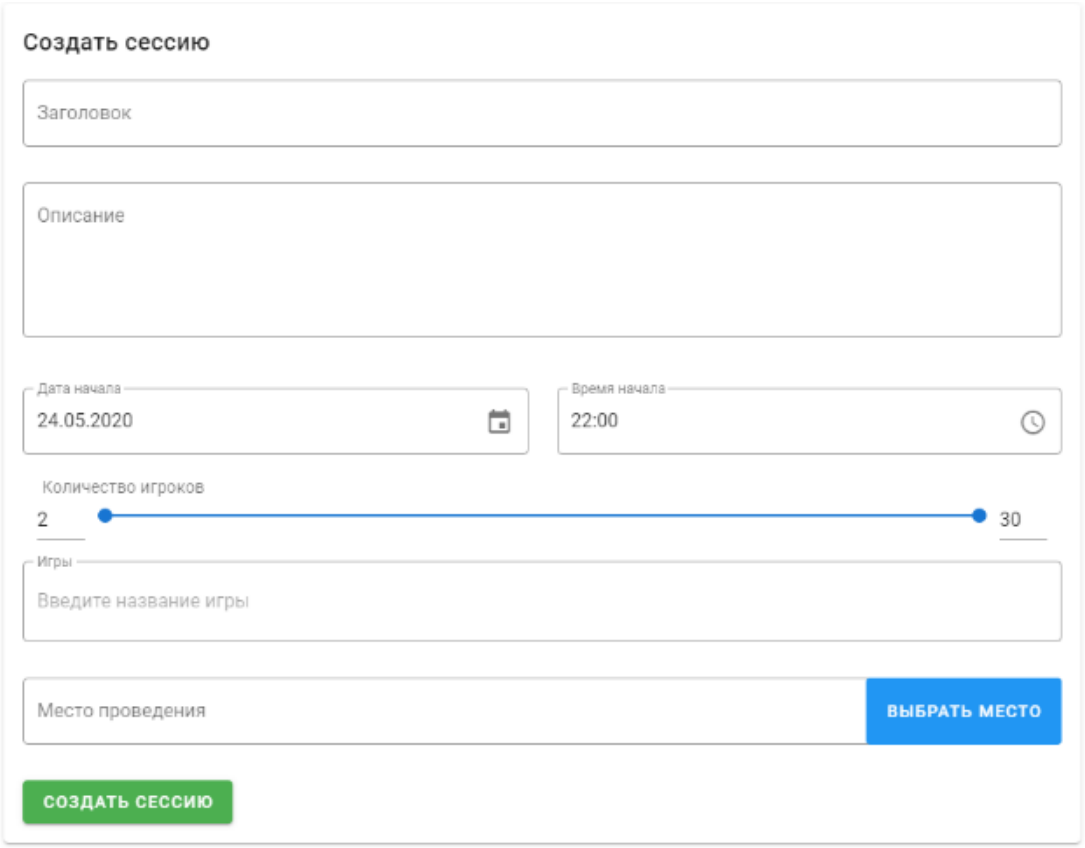

Рисунок 4.6 – Форма создания мероприятия

Игры для мероприятия выбираются при помощи текстового поля с автозаполнением, данные для него принимаются от Tesera API[14], которое предоставляется российским сайтом www.tesera.ru и содержит обширную базу настольных игр. Запросы к API совершаются только спустя некоторое время после остановки ввода пользователем, чтобы избежать слишком частых обращений к сервису, что достигается использованием функции debounce, которая гарантирует, что все вызовы будут игнорироваться до тех пор, пока они не прекратятся на определенный период времени.

Присутствует возможность выбора нескольких игр, выбор можно отменить нажатием крестика или удалением игры из текстового поля, например, клавишей backspace. Примеры работы с полем приведены на рисунках 4.7 и 4.8. Исходный код компонента приведен в листинге М.1 приложения М.

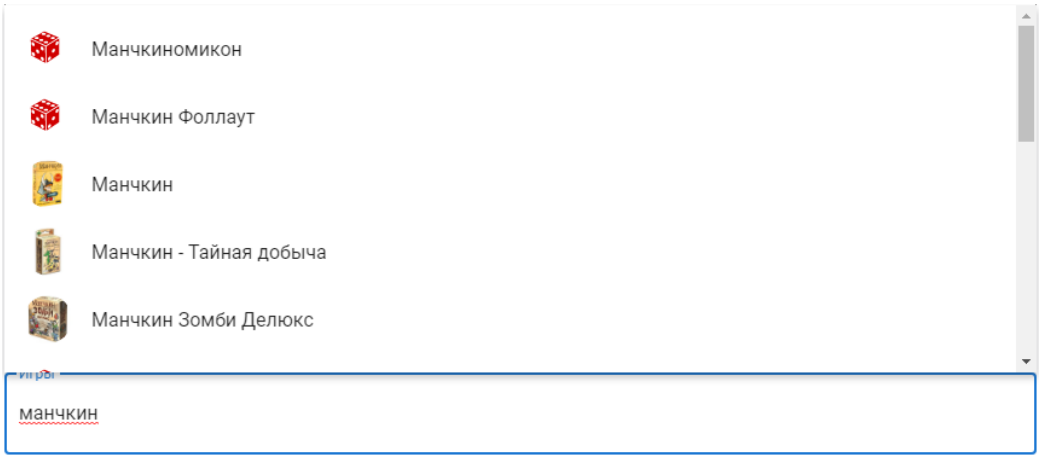

Рисунок 4.7 – Поиск игры по названию

Рисунок 4.8 – Добавление нескольких игр

Выбор места проведения мероприятия осуществляется при помощи диалогового окна с картой, вызвать которое можно нажатием кнопки в правой части соответствующего текстового поля. Адрес указывается нажатием на любой дом на карте или вводом в поисковую строку. Диалоговое окно с картой представлено на рисунке 4.9.

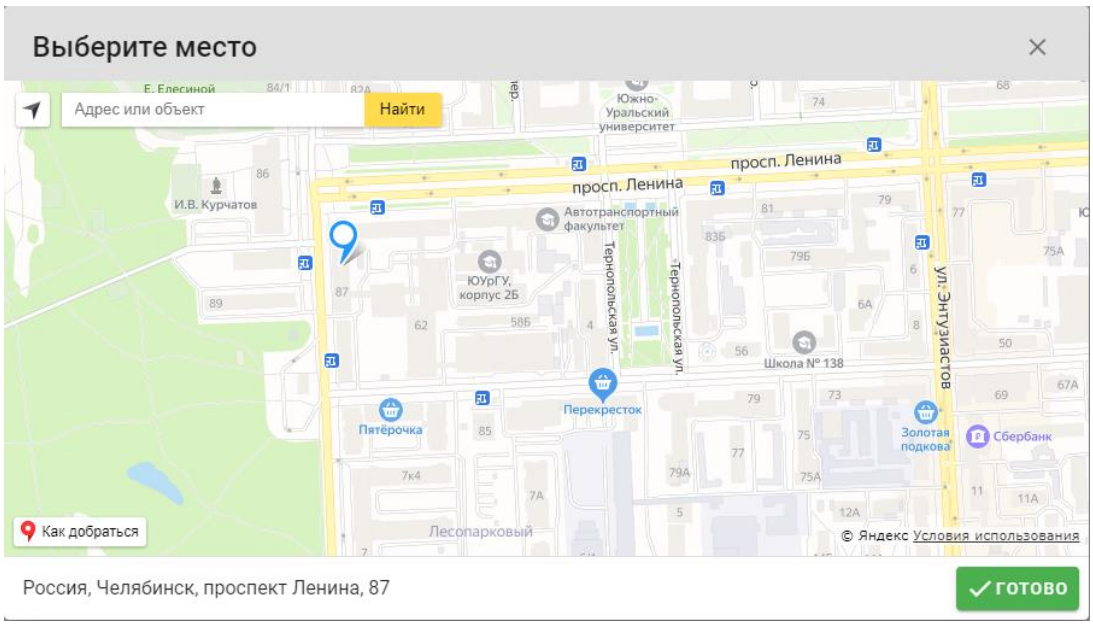

Рисунок 4.9 – Диалоговое окно выбора места проведения

Установка адреса производится за счет добавления обработчиков двух событий: нажатие курсором по карте и выбор пункта из выпадающего списка с результатами поиска. Оба обработчика устанавливают метку на соответствующую точку и проводят проверку корректности адреса. Код обработчиков представлены в листингах 4.1, 4.2, 4.3. Исходный код компонента приведен в листинге М.1 приложения М.

Листинг 4.1 – Обработчик события нажатия на результат поиска

```
this.search.events.add("resultselect", e => {
  this.search.getResult(e.get("index")).then(res => {
    this.point = res.geometry.getCoordinates();
    if (this.placemark) this.placemark.geometry.setCoordinates(this.point);
    else {
      this.placemark = new ymaps.Placemark(this.point);
      this.map.geoObjects.add(this.placemark);
    }
    this.setAddress(res);
  });
});
```

При нажатии на результат поиска вызывается функция библиотеки карт getResult, позволяющая получить более подробную информацию о выбранном геообъекте, аргументом которой является индекс результата. Далее на карту устанавливается отметка с координатами объекта, а его адрес проверяется на корректность в функции setAddress.

Листинг 4.2 – Обработчик события нажатия на карту

```
this.map.events.add("click", e => {
  this.point = e.get("coords");
  if (this.placemark) this.placemark.geometry.setCoordinates(this.point);
  else {
    this.placemark = new ymaps.Placemark(this.point);
    this.map.geoObjects.add(this.placemark);
  }
  ymaps.geocode(this.point).then(res => {
    this.setAddress(res.geoObjects.get(0));
  });
});
```

При нажатии на карту на местоположение курсора устанавливается метка, далее проводится обратное геокодирование координат нажатия. Полученный адрес аналогично проверяется на корректность в функции setAddress.

Листинг 4.3 – Вспомогательная функция setAddress

```
setAddress: function(geoObject) {
  if (geoObject) {
    if (
      geoObject.properties.get(
        "metaDataProperty.GeocoderMetaData.precision"
      ) == "exact") {
        this.isAddressValid = true;
        this.address = geoObject.getAddressLine();
    } else {
        this.isAddressValid = false;
        this.address = "Выберите дом";
    }
  }
}
```

Каждый геообъект Яндекс.Карт обладает свойством presicion (точность), его значение “exact” означает, что было выбрано здание. В таком случае адрес выводится в диалоговом окне, кнопка подтверждения становится активной, иначе выдается сообщение об ошибке.

После успешного создания мероприятия пользователь будет перенаправлен на страницу поиска мероприятий, которые можно вывести списком или показать на карте (рисунки 4.10 и 4.11). События в списке группированы по дате проведения и отсортированы по времени. Переключиться между режимами показа можно при помощи кнопок в верхней части страницы. Исходный код страницы приведен в листинге Б.1 приложения Б.

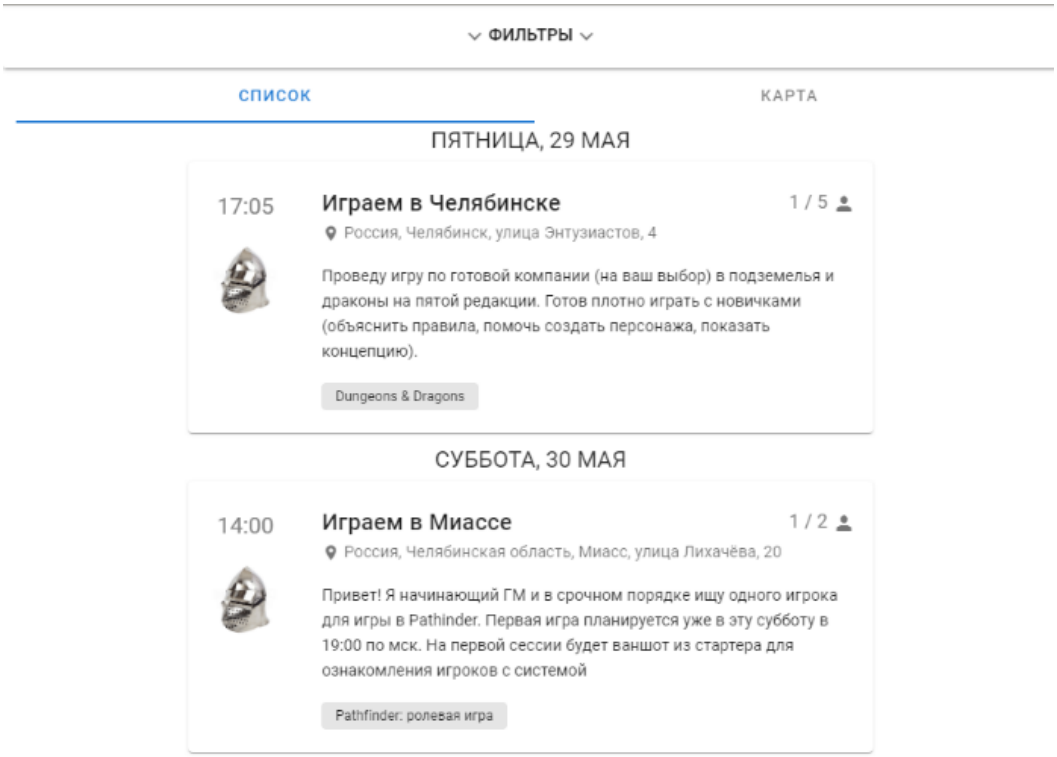

Рисунок 4.10 – Список мероприятий

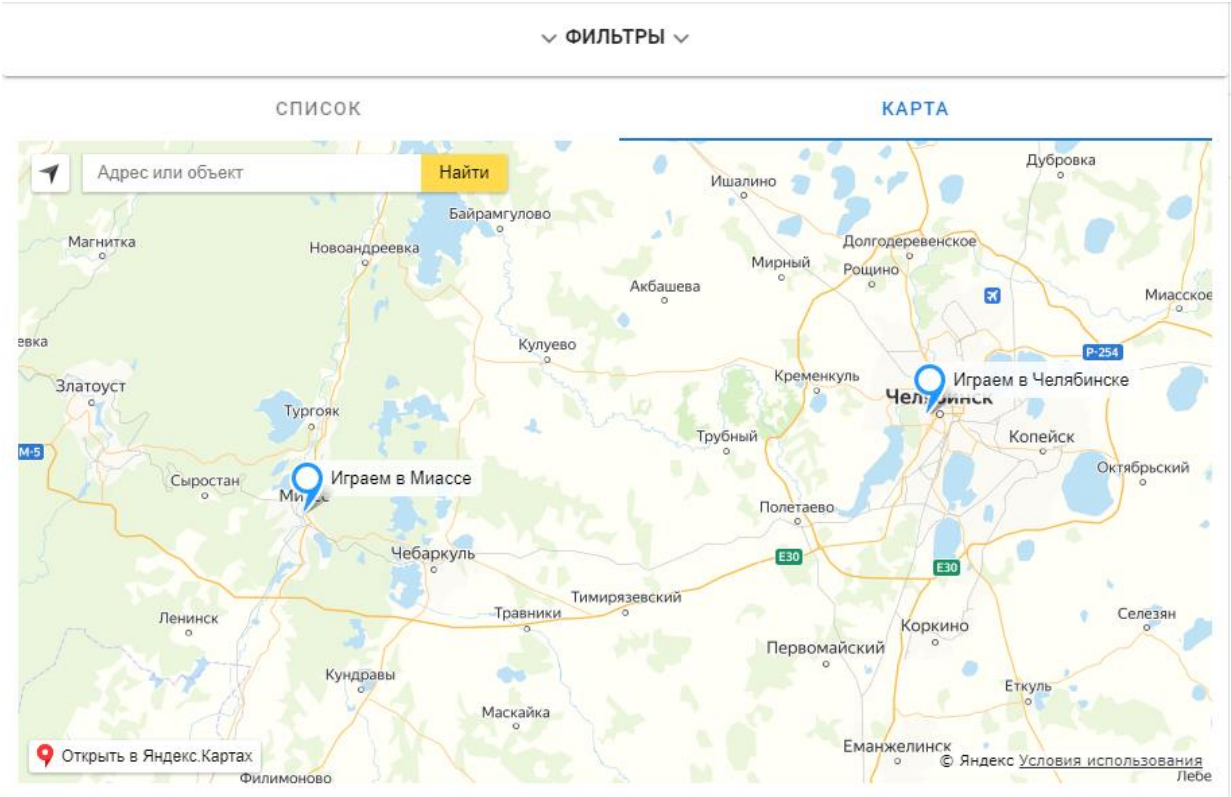

Рисунок 4.11 – Мероприятия на карте

Нажатие на отметку на карте откроет “балун”, содержащий краткую информацию о событии и ссылку него (рисунок 4.12).

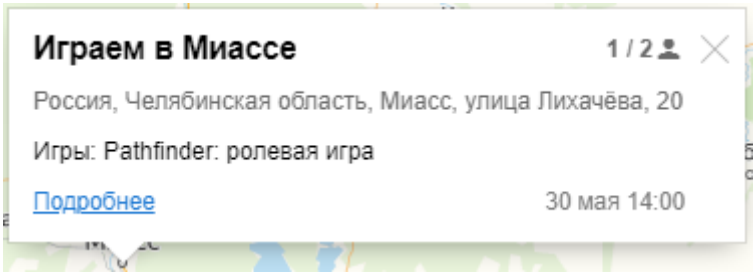

Рисунок 4.12 – Информация о мероприятии на карте

Присутствует фильтрация результатов (рисунок 4.13), открыть которую можно нажатием кнопки “фильтры”, присутствуют следующие критерии выбора:

– город, вокруг которого ведется поиск;
– радиус поиска в километрах;
– игры, доступные на мероприятии;
– промежуток дат проведения.

Выборка в базе данных производится при помощи динамического SQL запроса, критерии поиска сохраняются в параметрах URL, что позволяет восстановить фильтры при обновлении страницы. Фрагмент исходного кода, отвечающий за динамический запрос к базе данных представлен в листинге 4.4.

Листинг 4.4 – Запрос для получения списка мероприятий

```
getEvents: function (query) {
    return new Promise((resolve, reject) => {
      let where = mysql.format("WHERE events.datetime BETWEEN ? AND ?", [
        query.start_date
          ? moment(query.start_date).toDate()
          : moment().startOf("day").toDate(),
        query.end_date
          ? moment(query.end_date).toDate()
          : moment().add(7, "days").startOf("day").toDate(),
      ]);
```

Продолжение листинга 4.4

```
      if (query.games) {
        where += mysql.format(
          " AND events.id IN (SELECT event_id FROM event_games WHERE gametitle IN (?))",
          [query.games]
        );
      }
      if (query.point) {
        where += mysql.format(" AND ST_Distance_Sphere(events.point, ?) < ?", [
          mysql.raw(`POINT(${query.point[1]}, ${query.point[0]})`),
          (query.radius || 50) * 1000,
        ]);
      }
      pool.query(
        `SELECT events.*, profiles.name, profiles.avatar, group_concat(DISTINCT event_gam
es.gametitle) as games, COUNT(DISTINCT events_users.user_id) as player_count FROM events
JOIN profiles ON events.owner_id = profiles.user_id LEFT JOIN event_games ON events.id =
event_games.event_id LEFT JOIN events_users ON events.id = events_users.event_id ${where}
 GROUP BY events.id ORDER BY unix_timestamp(events.datetime) ASC`,
        (error, results, fields) => {
          if (error) {
            reject(error);
          }
          if (results) {
            resolve(results);
          }
        }
      );
    });
  }
```

Функция проверяет наличие параметров в URL и в случае их обнаружения добавляет соответствующее условие в запрос. Дистанция между выбранной пользователем точкой и местами проведения мероприятий определяется при помощи функции MySQL ST_Distance_Sphere, которая вычисляет расстояние между двумя точками (тип данных POINT) в метрах с учетом формы Земли.

Во избежание SQL инъекций, пользовательские данные подставляются через специальную функцию (mysql.format). При отсутствии в параметрах промежутка дат используются значения по умолчанию.

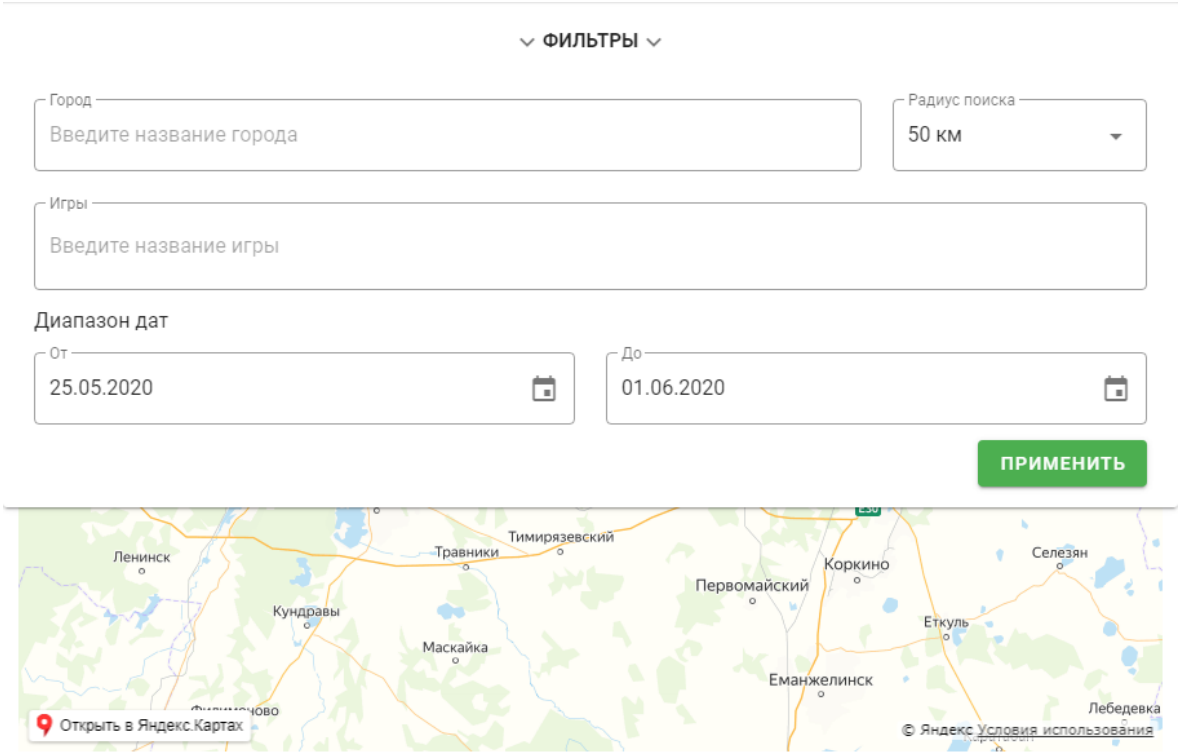

Рисунок 4.13 – Фильтрация мероприятий

На рисунке 4.14 представлена страница мероприятия, которая содержит всю информацию о событии, ссылку на подробный список участников (рисунок 4.15), кнопку “присоединиться” и комментарии, которые могут оставлять только участники. Исходный код страницы приведен в листинге В.1 приложения В.

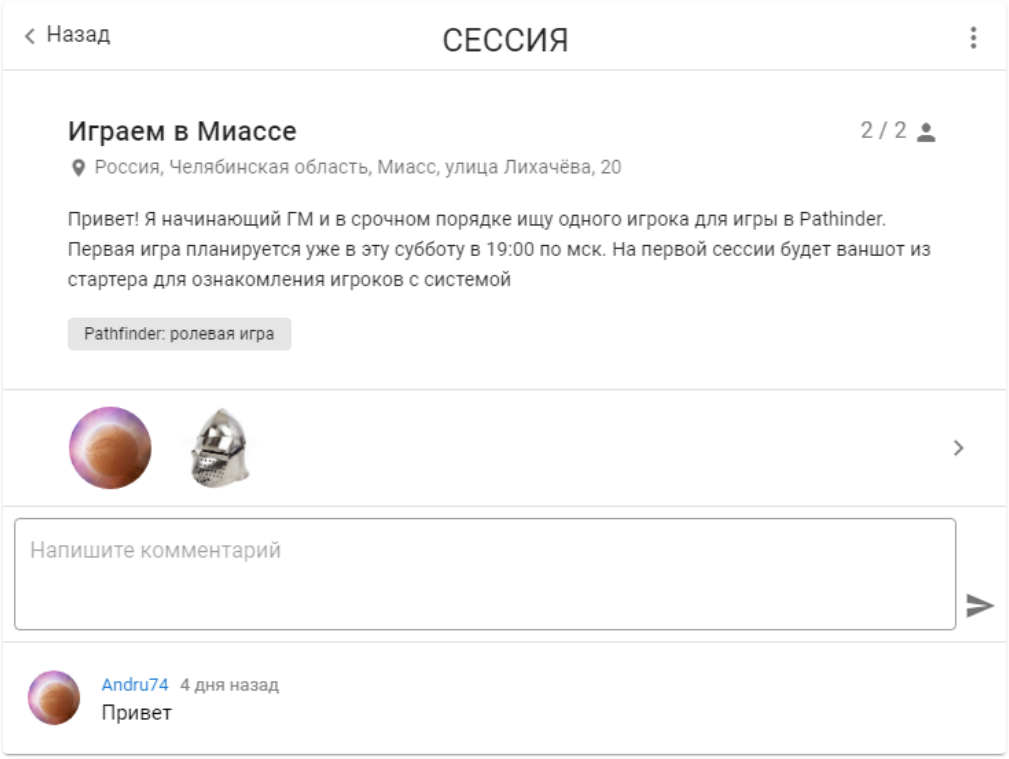

Рисунок 4.14 – Страница мероприятия

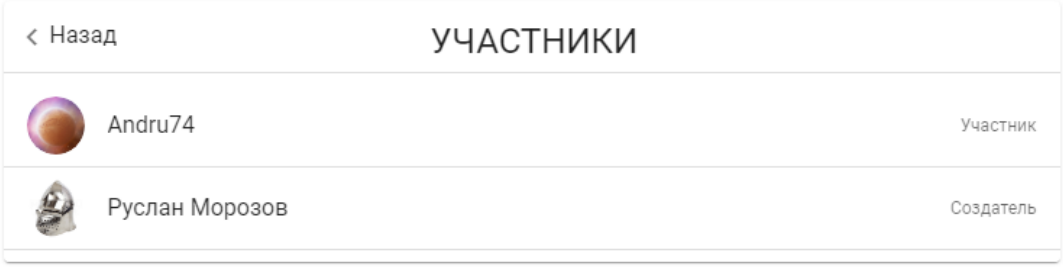

Рисунок 4.15 – Список участников

Пользователи, находящиеся на странице, получают комментарии в реальном времени за счет применения протокола WebSocket. При использовании протокола HTTP, чтобы получить данные, клиент должен каждый раз отправлять запрос, что может вызвать излишнюю нагрузку на сервер. С другой стороны, WebSocket устанавливает постоянное соединение, поэтому данные с сервера могут поступить в любой момент после подключения, а не запросу клиента.

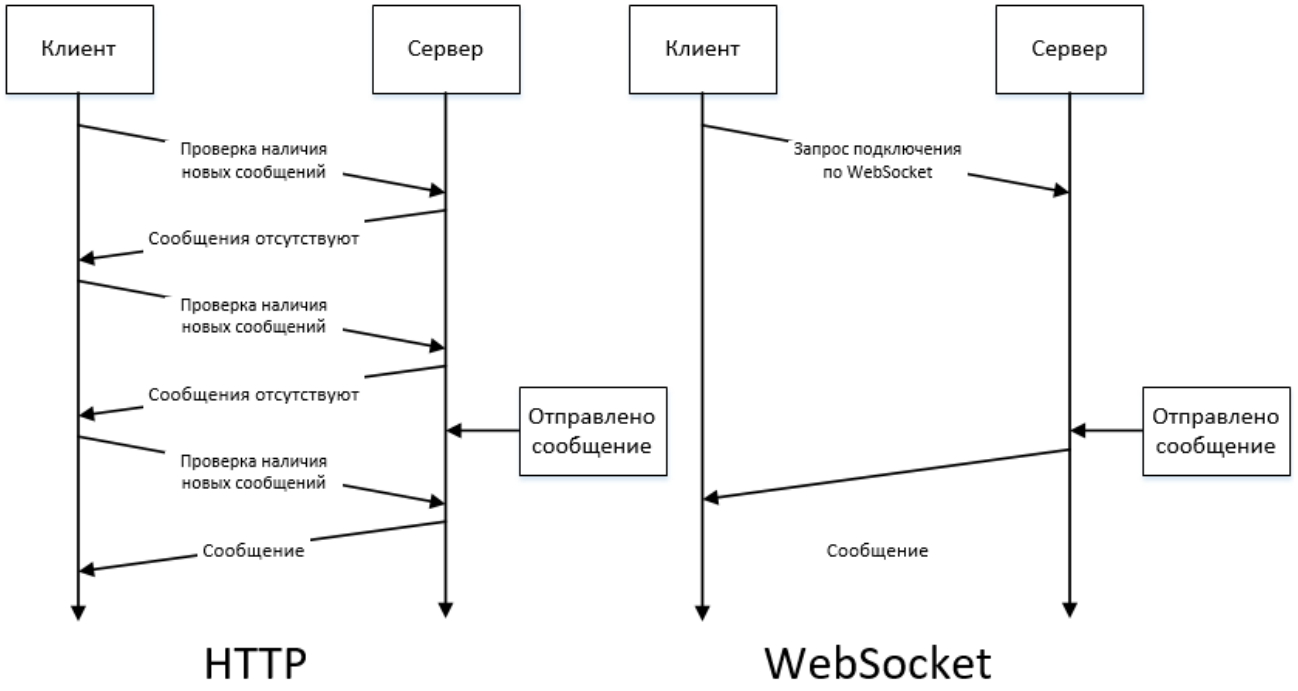

Рисунок 4.16 – Сравнение работы протоколов HTTP и WebSocket

Работа с протоколом осуществляется при помощи библиотеки Socket.IO: когда пользователь получает доступ к странице мероприятия, его браузер отправляет запрос на сервер с указанием идентификатора события, сервер отвечает на его запрос и устанавливает соединение, назначая пользователя в соответствующую мероприятию “комнату”.

В дальнейшем, при получении комментария мероприятия, сервер пересылает его всем подключившимся к “комнате” пользователям.

4.4. ПРОФИЛЬ ПОЛЬЗОВАТЕЛЯ

Страница пользователя, представленная на рисунке 4.17, содержит основную информацию, расписание пользователя и ссылки на его аккаунты в социальных сетях. Исходный код страницы приведен в листинге Е.1 приложения Е.

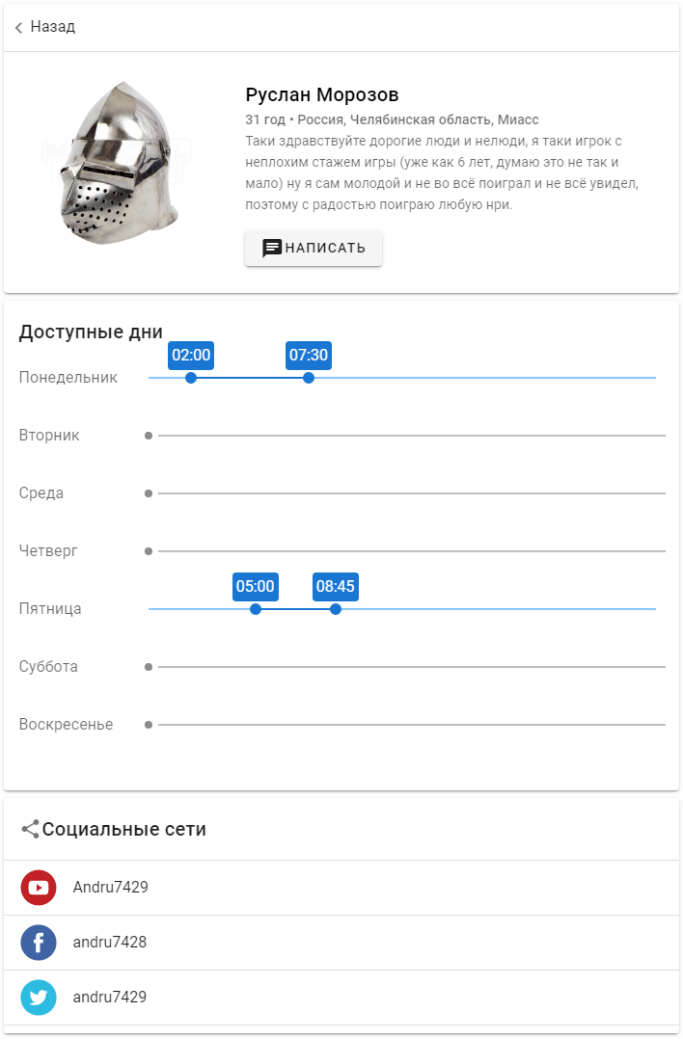

Рисунок 4.18 – Страница пользователя

Страница редактирования профиля представлена на рисунке 4.20. Исходный код страницы приведен в листинге Ж.1 приложения Ж. Расписание пользователя устанавливается при помощи флажков и двойных ползунков.

Изображения профиля можно изменить нажатием на нём, расширение отправленного файлом пользователя производится на сервере.

Добавление аккаунта в социальных сетях осуществляется OAuth авторизации, которая совершается по следующим шагам [15]:

1. При нажатии кнопки добавления аккаунта пользователя перенаправляет на окно авторизации соответствующей социальной сети.
2. При успешной авторизации пользователя его перенаправляет на заданный маршрут приложения со специальным кодом в параметрах.
3. Сервер приложения совершает запрос к социальной сети с полученным кодом и в результате получает токен авторизации пользователя, при помощи которого можно получить необходимую информацию, например, имя, фамилию или идентификатор.

При успешной авторизации ссылка на аккаунт добавляется в профиль пользователя. На текущем этапе разработки данная функция реализована только для социальной сети ВКонтакте.

Схема OAuth авторизации представлена на рисунке 4.19.

Рисунок 4.19 – Описание процесса авторизации

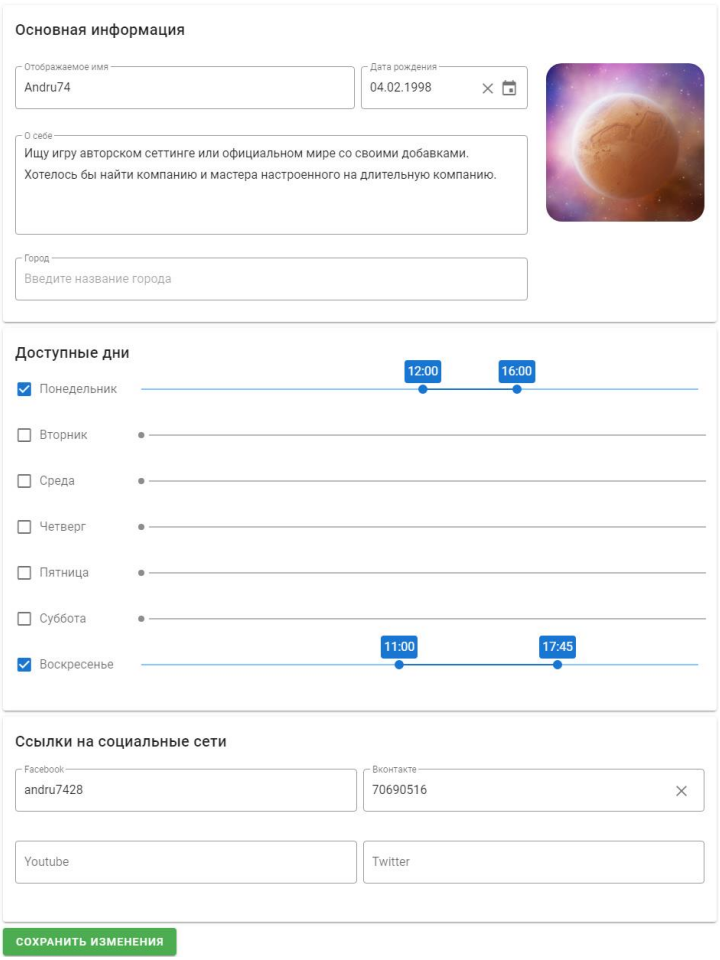

Рисунок 4.20 – Страница редактирования профиля

На рисунке 4.21 представлена страница поиска пользователей, результаты можно фильтровать по местоположению игроков. Исходный код страницы приведен в листинге Д.1 приложения Д.

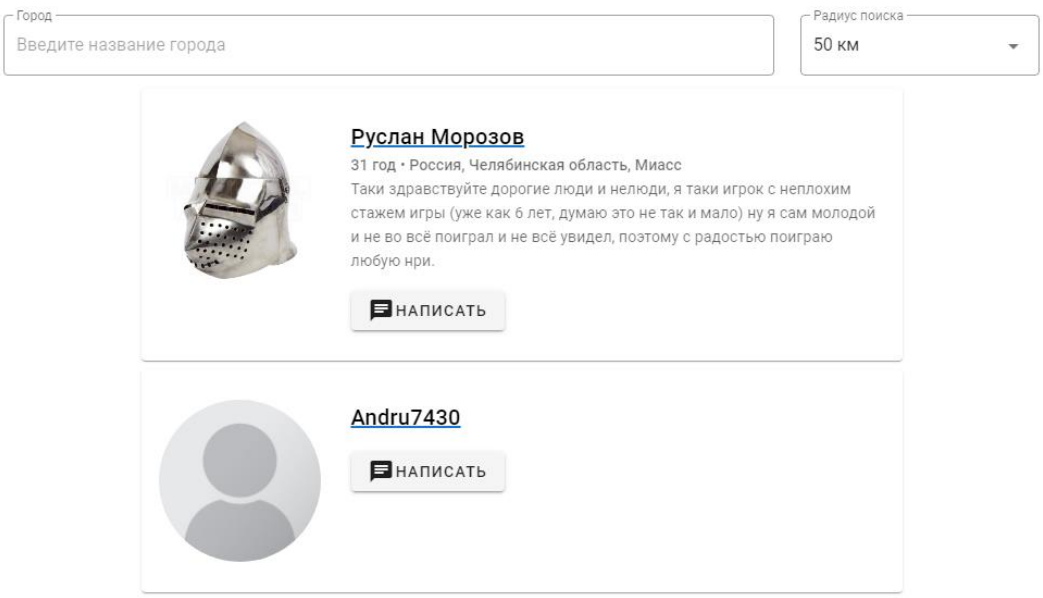

Рисунок 4.21 – Страница поиска пользователей

4.5. ЛИЧНЫЕ СООБЩЕНИЯ ПОЛЬЗОВАТЕЛЯ

На рисунке 4.22 представлен список диалогов пользователя. Для устройств с большой шириной экрана функционал этой страницы дублируется в правой боковой панели. Нажатие на пункт в списке направит пользователя на страницу с диалогов.

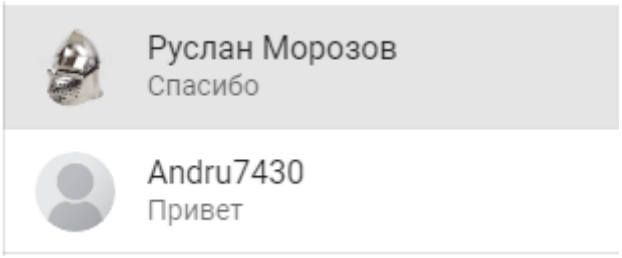

Рисунок 4.22 – Список диалогов пользователя

Страница диалога представлена на рисунке 4.23. Пользователь получает сообщения в реальном времени за счет использования протокола WebSocket. Исходный код страницы приведен в листинге К.1 приложения К.

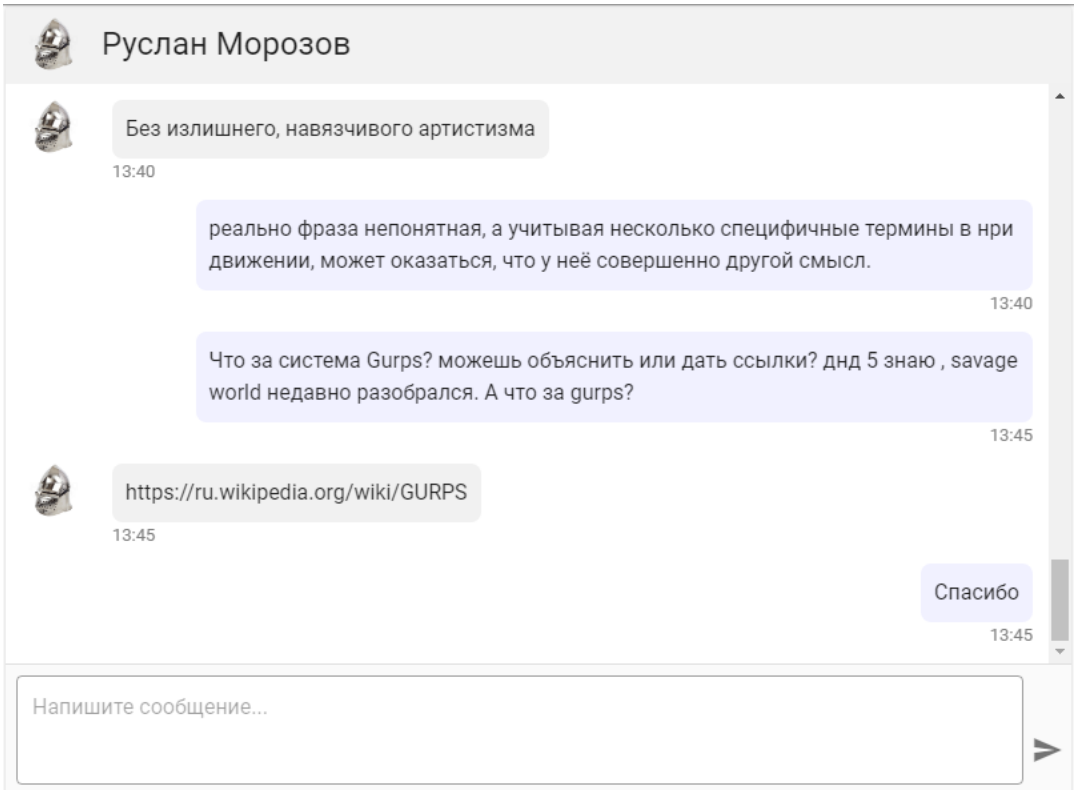

Рисунок 4.23 – Страница диалога

5. ТЕСТИРОВАНИЕ

5.1. ПРОВЕДЕНИЕ ПРОЦЕДУРЫ ТЕСТИРОВАНИЯ

В рамках дипломного проекта было проведено функциональное тестирование реализованной системы. Ниже приведён перечень тестов.

1. Тестирование форм авторизации и регистрации.

На рисунках 5.1, 5.2, 5.3 приведены примеры проверки данных, указанных пользователем при авторизации.

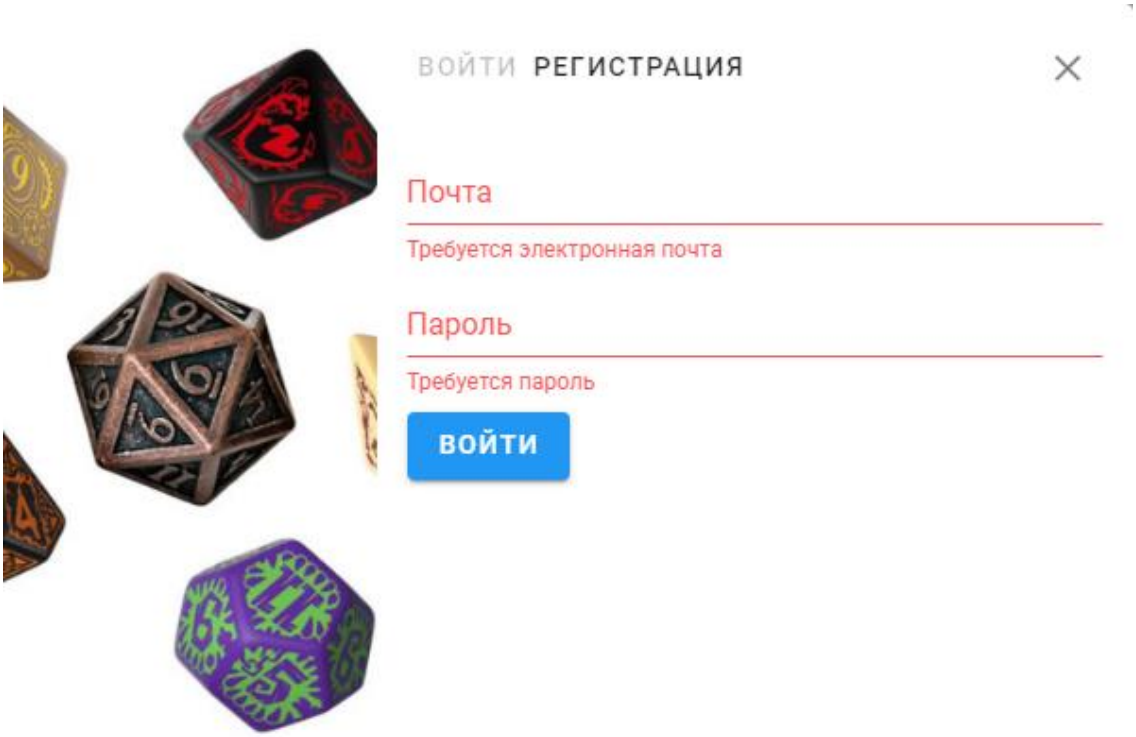

Рисунок 5.1 – Валидация формы авторизации, не заполнены обязательные поля

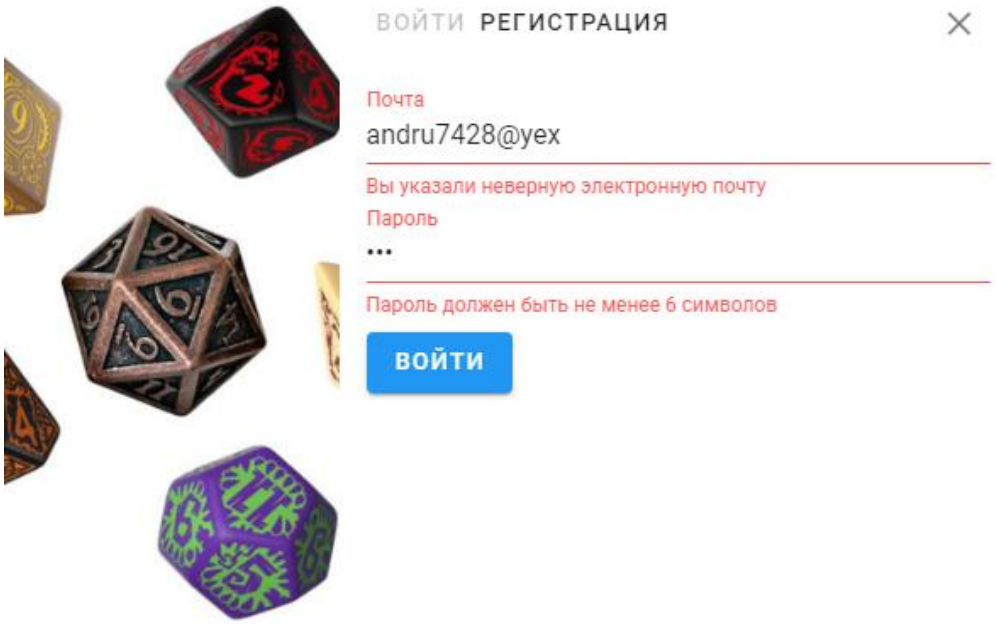

Рисунок 5.2 – Валидация формы авторизации, введены некорректные данные

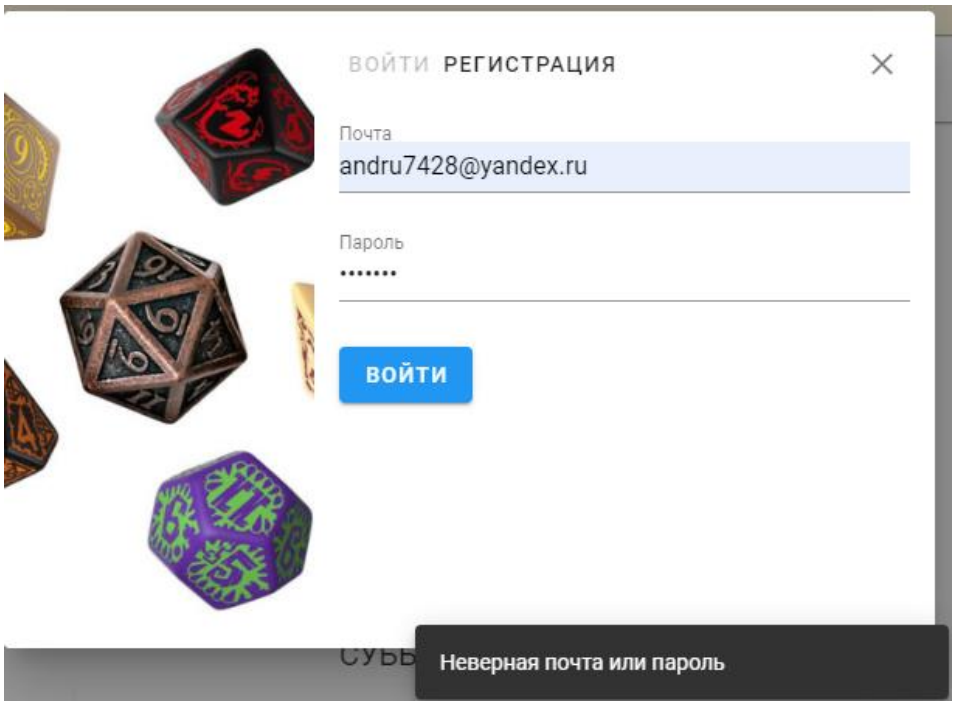

Рисунок 5.3 – Проверка данных, отправленных пользователем, введены неверные данные

На рисунках 5.4, 5.5, 5.6 приведены примеры проверки данных, указанных пользователем при регистрации.

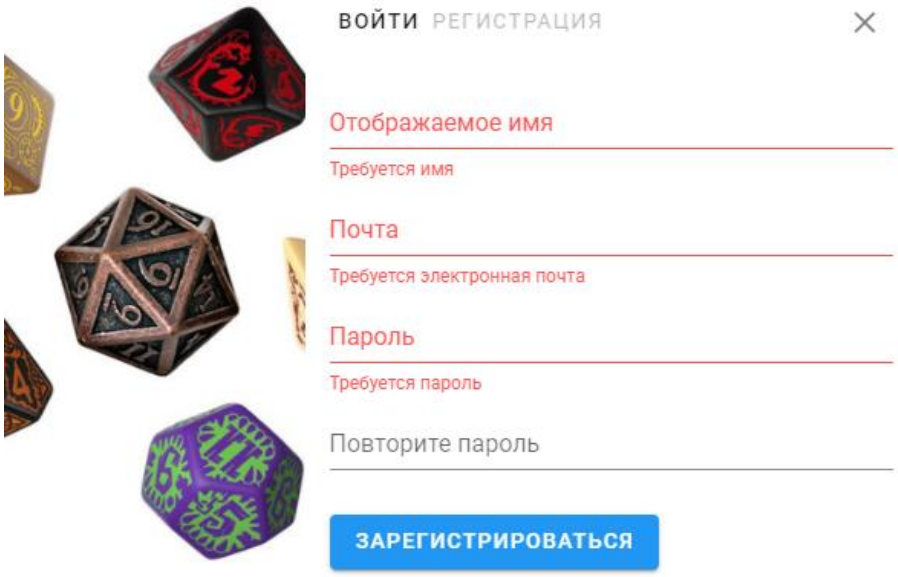

Рисунок 5.4 – Валидация формы регистрации, не заполнены обязательные поля

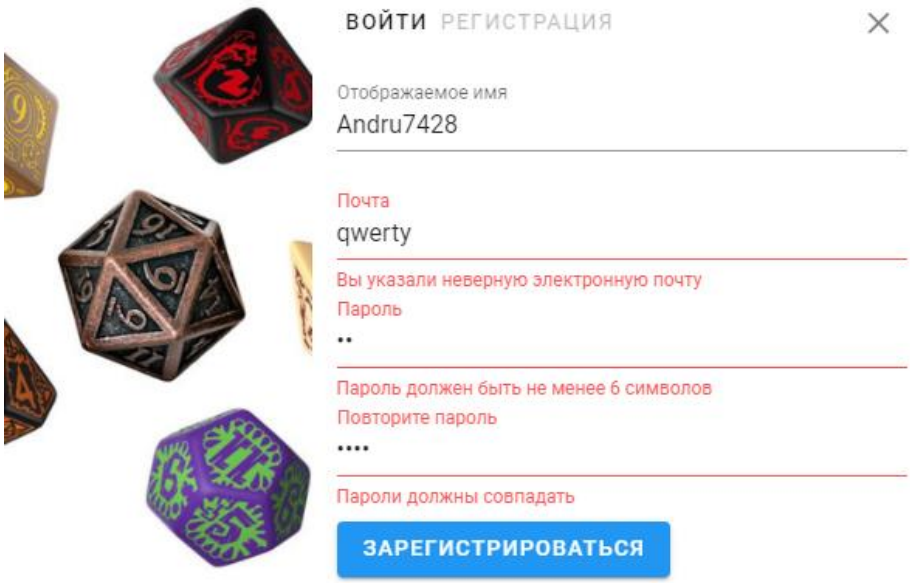

Рисунок 5.5 – Валидация формы регистрации, введены некорректные данные

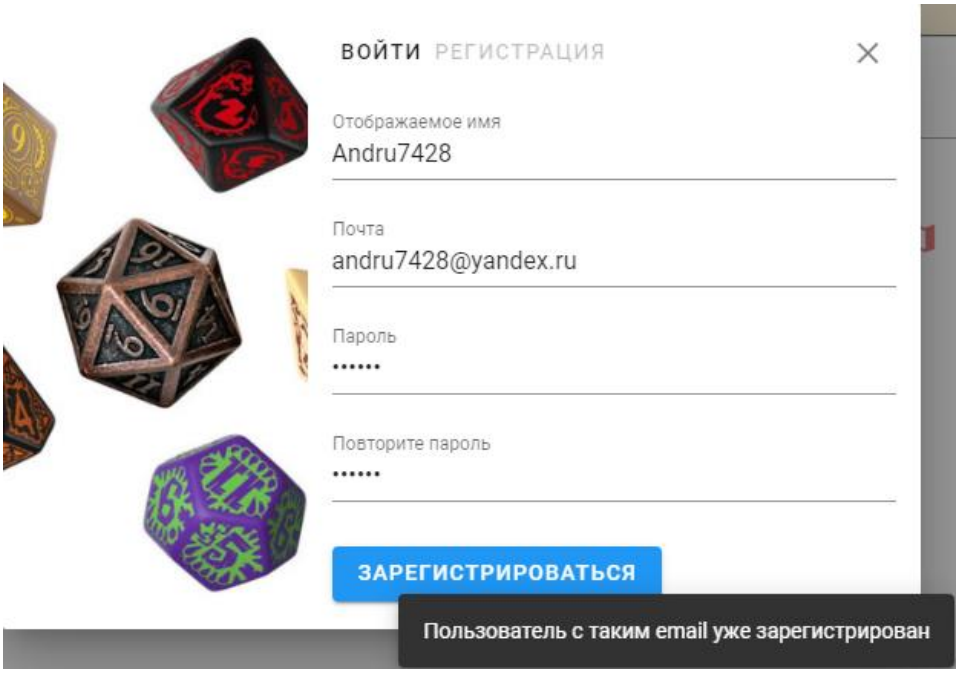

Рисунок 5.6 – Проверка данных, отправленных пользователем, введены дублирующие данные

2. Тестирование формы создания мероприятия.

Проверка введенных данных при создании мероприятия представлена на рисунке 5.7.

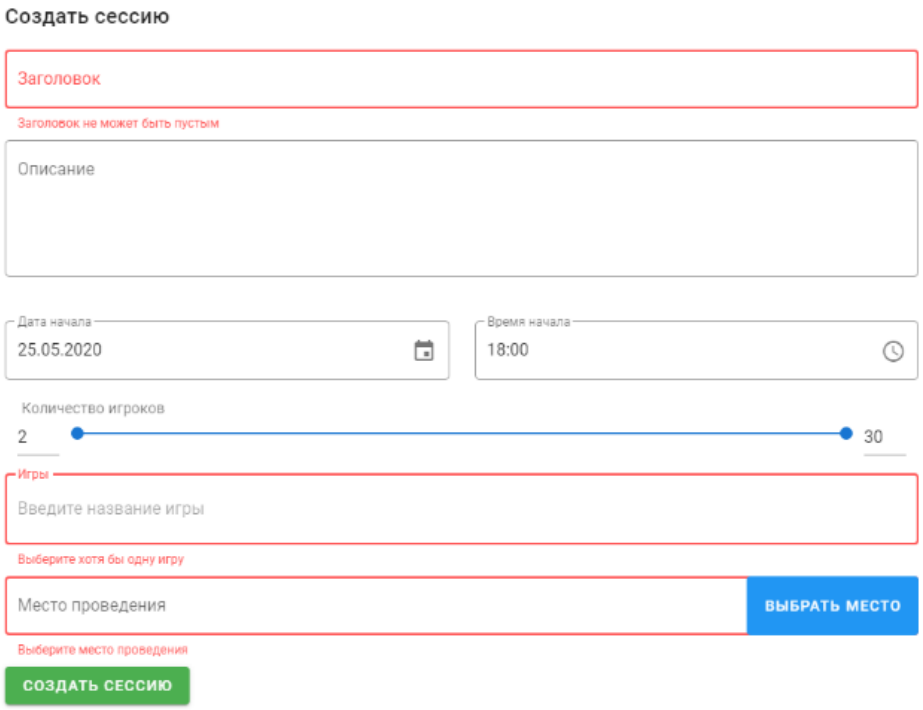

Рисунок 5.7 – Валидация данных формы создания мероприятий

3. Тестирование формы редактирования профиля пользователя.

Введенные пользователем данные представлены на рисунке 5.8.

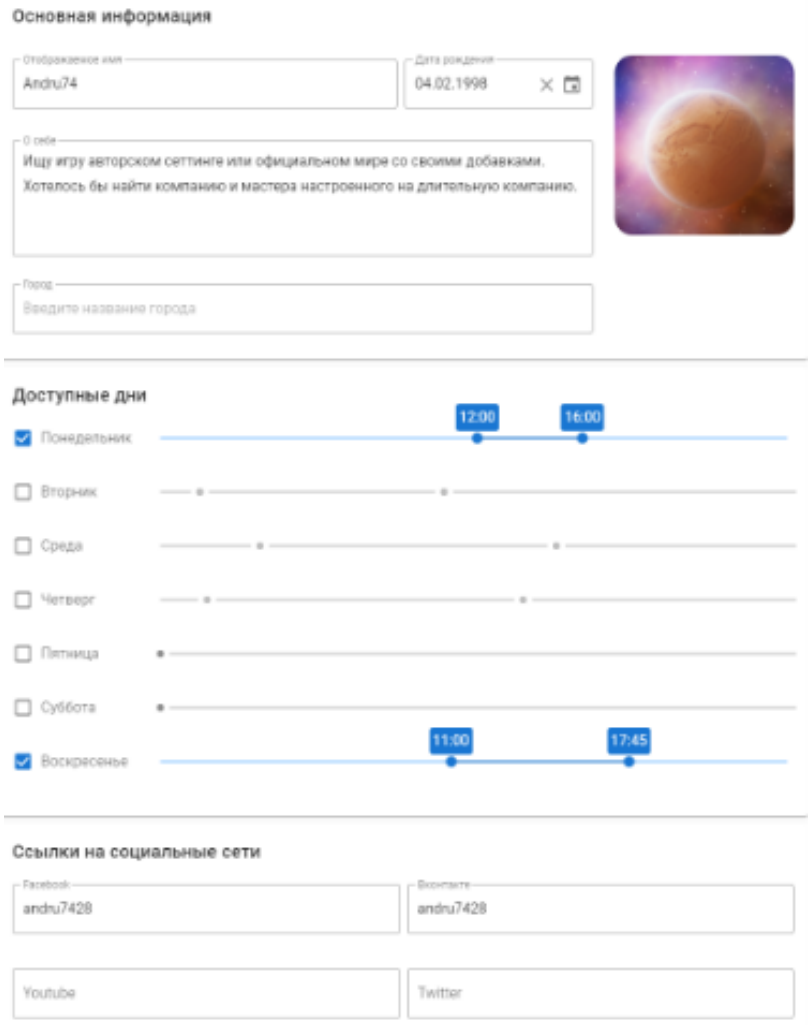

Рисунок 5.8 – Введенные пользователем данные при редактировании профиля

На рисунке 5.9 представлен профиль пользователя после редактирования.

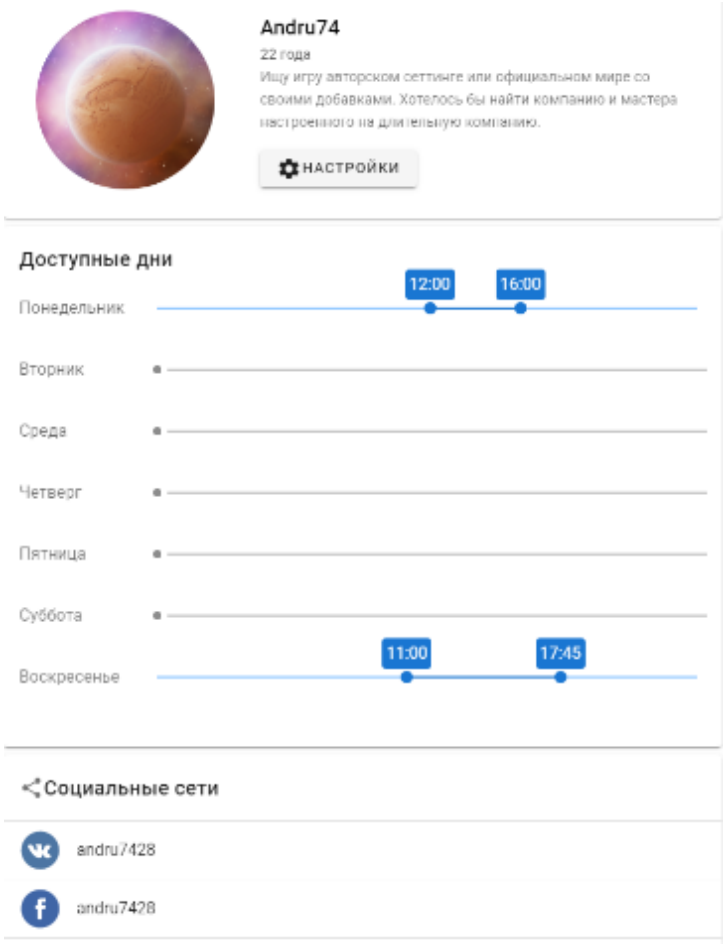

Рисунок 5.9 – Профиль пользователя после редактирования

4. Тестирование фильтрации мероприятий.

Примеры фильтрации мероприятий приведены на рисунках 5.10, 5.11, 5.12, 5.13.

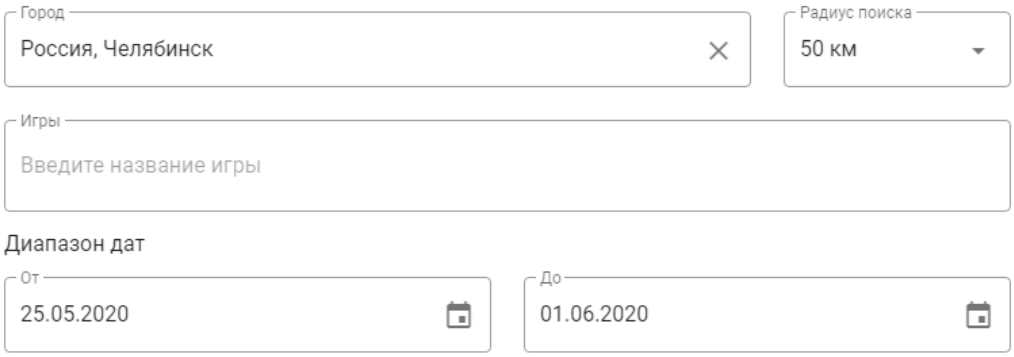

Рисунок 5.10 – Установленные критерии поиска

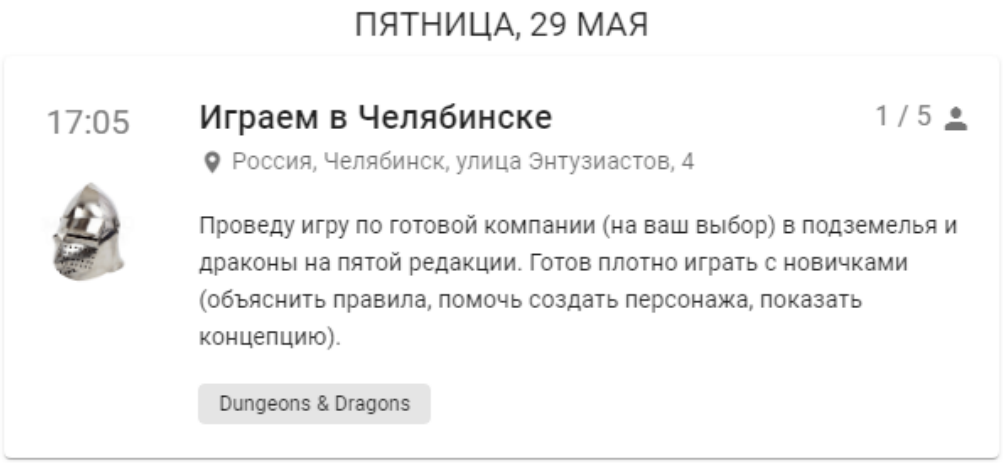

Рисунок 5.11 – Результаты поиска

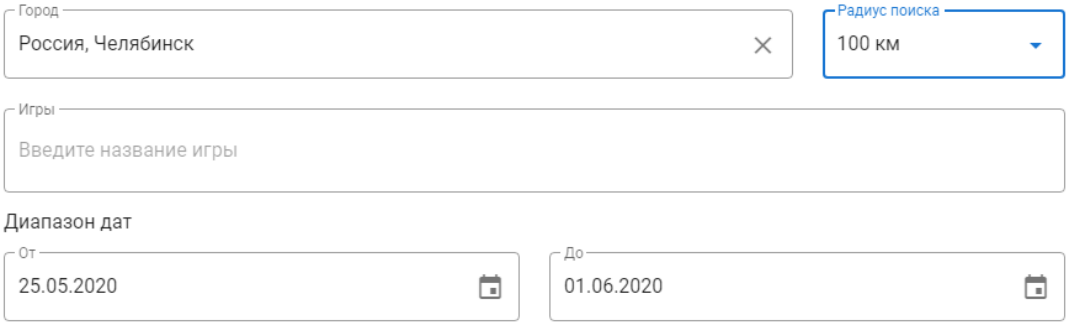

Рисунок 5.12 – Новые критерии поиска

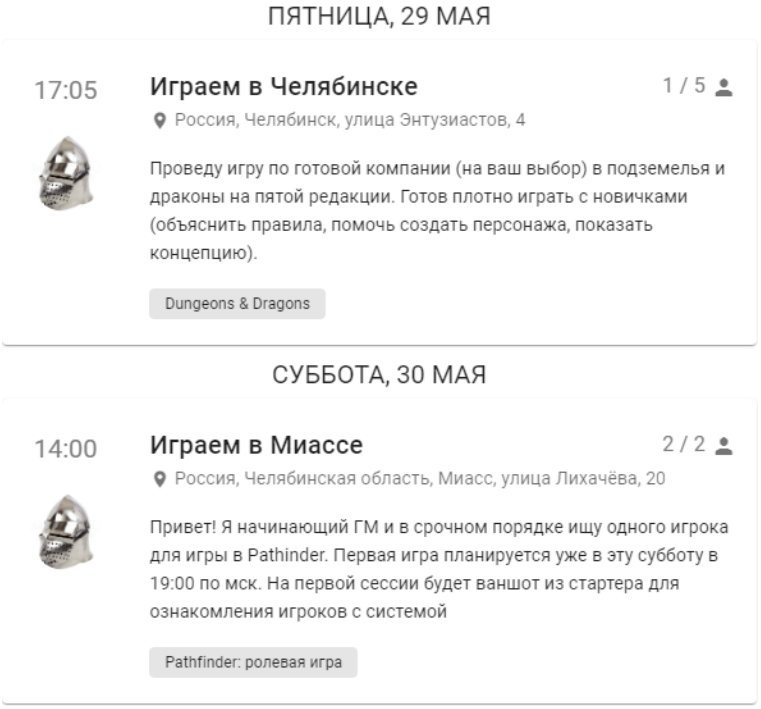

Рисунок 5.13 – Новые результаты поиска

По полученным результатам тестирования поиска видно, что фильтрация по расстоянию работает корректно (расстояние между Челябинском и Миассом 83 км по прямой).

5. Тестирование отправки личных сообщений.

Примеры отправки личного сообщения и получения его в реальном времени приведены на рисунках 5.14, 5.15.

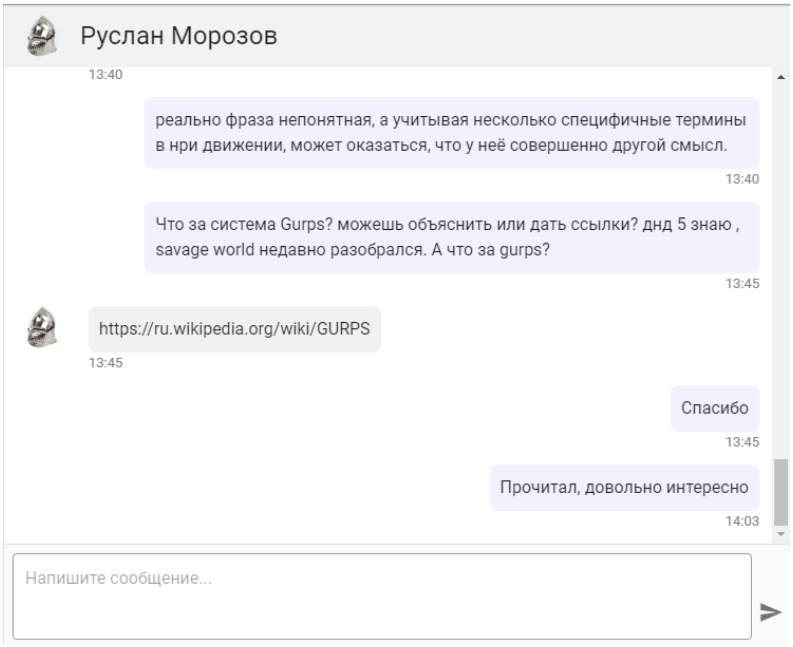

Рисунок 5.14 – Отправка личного сообщения

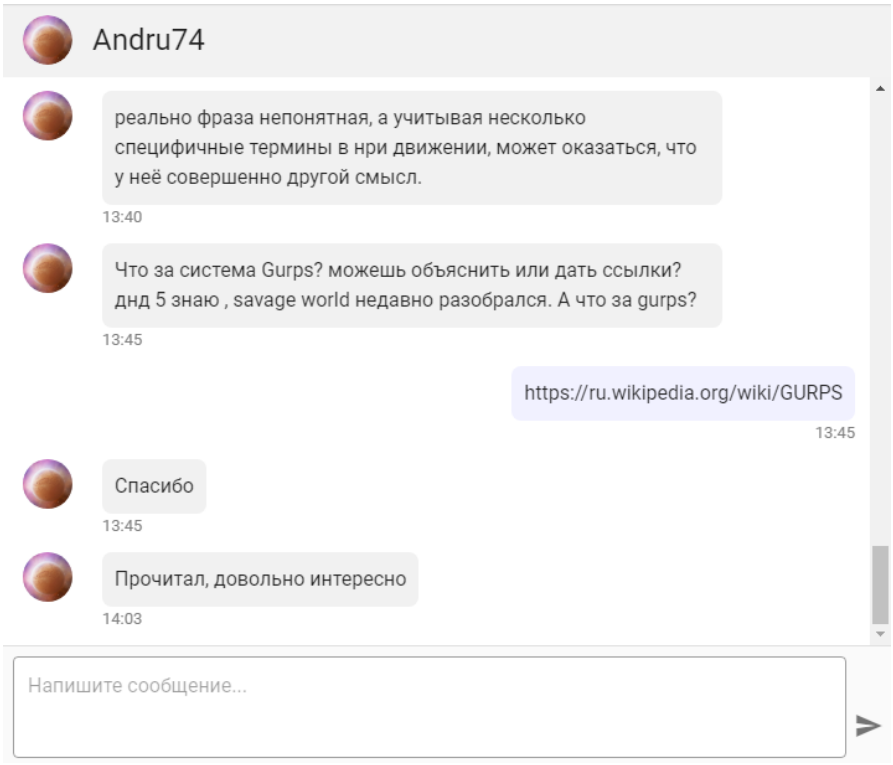

Рисунок 5.15 – Получение личного сообщения

6. ЗАКЛЮЧЕНИЕ

В рамках дипломного проекта было выполнено следующее:

1. Проведен анализ предметной области и аналогичных разработок;
2. Проанализированы и выбраны технические решения для реализации проекта;
3. Определены функциональные и нефункциональные требования к приложению;
4. Спроектирована архитектура разрабатываемого решения;
5. Разработана база данных;
6. Реализовано и протестировано программное обеспечение, реализующее поставленную задачу.

В разработанном веб-приложении реализованы следующие функции:

1. Создание мероприятий с указанием места проведения на карте и использованием базы данных настольных игр;
2. Поиск мероприятий с фильтрацией по месту проведения и играм;
3. Возможность вывода результатов поиска на карту;
4. Поиск игроков с фильтрацией по городам;
5. Комментирование мероприятий с получением сообщений в реальном времени;
6. Система личных сообщений в реальном времени;

В дальнейшем необходимо реализовать систему администрирования, после чего системы будет готова к эксплуатации.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Roll for Group - Meet up and get your game group rolling – https://www.rollforgroup.com/. (Дата обращения: 01.03.2020).
2. GameTree – https://gametree.me/ru/. (Дата обращения: 01.03.2020).
3. GameFor | Connecting Tabletop Gamers Mobile App – https://iamgamefor.com/. (Дата обращения: 01.03.2020).
4. Meetup – https://www.meetup.com/. (Дата обращения 20.04.2020).
5. Что такое база данных | Oracle Россия и СНГ – https://www.oracle.com/ru/database/what-is-database.html. (Дата обращения: 01.03.2020).
6. What Is a Relational Database | Oracle – https://www.oracle.com/database/what-is-a-relational-database/. (Дата обращения: 01.03.2020).
7. Что такое NoSQL | Нереляционные базы данных, модели данных с гибкой схемой | AWS – https://aws.amazon.com/ru/nosql/. (Дата обращения: 01.03.2020).
8. SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems | DigitalOcean – https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems. (Дата обращения: 01.03.2020).
9. MySQL :: MySQL 8.0 Reference Manual :: 1 General Information – https://dev.mysql.com/doc/refman/8.0/en/introduction.html. (Дата обращения: 01.03.2020).

10. Usage Statistics and Market Share of Server-side Programming Languages for Websites, March 2020 – https://w3techs.com/technologies/overview/programming_language. (Дата обращения: 01.03.2020).
11. Всё что вам нужно знать о Node.js / Хабр – https://habr.com/ru/post/460661/.
12. What Are the Best Node.js Frameworks in 2019 and Why? Top 5 – https://medium.com/@OPTASY.com/what-are-the-best-node-js-frameworks-in-2019-and-why-top-5-4434770d2187. (Дата обращения: 01.03.2020).
13. Обзор наиболее популярных картографических сервисов, предоставляющих API для разработчиков – https://novainfo.ru/article/13853. (Дата обращения: 01.03.2020).
14. Tesera API – https://api.tesera.ru/help/index.html. (Дата обращения: 20.04.2020).
15. Введение в OAuth2 – https://www.digitalocean.com/community/tutorials/oauth-2-ru. (Дата обращения: 20.04.2020)